MAITS SOFTWARE

Proyecto de fin de curso: **MINERVA**

Sistema de gestión de bedelía

Empresa: MAITS Software

Integrantes: Ignacio Rodríguez, Agustina Martínez

Curso: Bachillerato de Informática

Institución: UTU Canelones

Año: 2016

Docentes y Asignaturas:

- Formación Empresarial: Lorena Suárez.
- Análisis y Diseño de Aplicaciones: Daniel Carrasco.
- Taller de Mantenimiento: Daniel Carrasco.
- Base de Datos: Victoria Bolsi.
- Sistemas Operativos: Gustavo Caraballo.
- Proyecto: Ignacio Olveira.
- Programación: Ignacio Olveira.

Información del documento

Organización	MAITS Software
Proyecto	Minerva
Delegado de proyecto	Ignacio Rodríguez
Integrantes del grupo	Agustina Martínez, Ignacio Rodríguez
Documento	Cuarta entrega
Nombre del archivo	Cuarta entrega MAITS Software
Fecha de creación	17-11-2016

Histórico de versiones

V1	10-09-2016
V2	22-10-2016
V3	17-11-2016

Distribución

	Organización	Nombre	# de copias	Comentario
V1	Escuela Técnica Canelones	Segunda entrega MAITS	1	Para segunda entrega
V2	Escuela Técnica Canelones	Tercera entrega MAITS	1	Para tercera entrega
V3	Escuela Técnica Canelones	Cuarta entrega MAITS	1	Para cuarta entrega

1. Tabla de contenido

2. Nuestra Empresa

2.1 Misión

Nuestra misión como empresa dedicada al desarrollo de software es ofrecer soluciones puntuales e innovadoras, atención personalizada y rápida respuesta a las contingencias del cliente, mediante un equipo técnico profesional y comprometido.

2.2 Visión

Nuestra visión es ser la empresa que siempre se adecúe mejor a las necesidades de nuestros clientes, convirtiéndonos en sus socios estratégicos. Pretendemos ser referentes dentro del sector de desarrollo de software, y para ello abarcaremos todos los servicios que ofrecemos actualmente, e incrementando los que vayan surgiendo debido a la necesidad de cambio provocado por los avances tecnológicos.

2.3 Objetivos generales

- Ser la mejor empresa de desarrollo de software.
- Ofrecer a nuestros clientes atención personalizada para satisfacer totalmente sus necesidades.
- Tener la mejor relación calidad-precio.
- Facilitar el trabajo de las personas que actualmente realizan las mismas funciones pero de forma manual.
- Estar al tanto de las sugerencias del cliente.
- Desarrollar siempre los mejores productos.
- Estar al tanto de los avances tecnológicos.
- Usar de la forma más eficiente los recursos de la empresa.

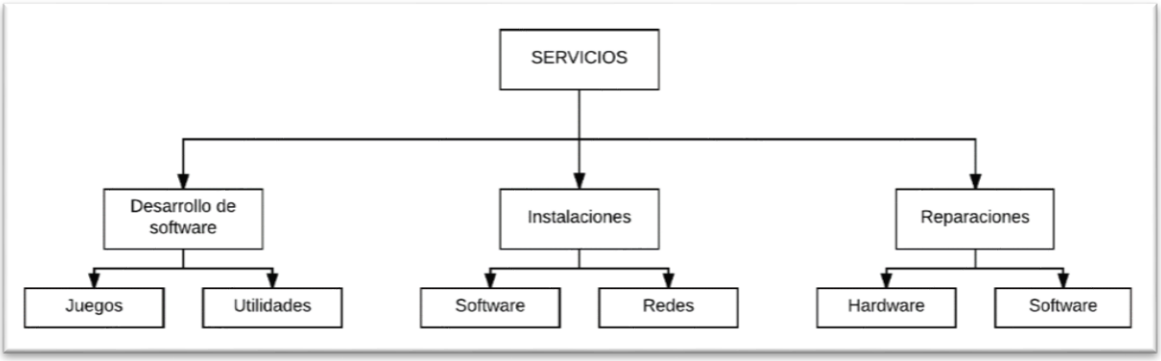
2.4 Objetivos específicos

- Duplicar la cantidad de clientes antes de fin de año.
- Adquirir nuevos equipos antes del segundo trimestre del próximo año.

- Aumentar el personal el próximo semestre.
- Aumentar las ventas anuales un 20%.
- Aumentar la participación en el mercado un 25% el próximo trimestre.

2.5 Estructura organizativa

MAITS Software tiene definida su estructura organizativa en base a los servicios que le brinda al cliente.



2.6 Recursos

Los recursos son necesarios para optimizar el rendimiento de la empresa. Para tener un buen posicionamiento en el mercado, es necesario mantenerlos en las mejores condiciones posibles.

2.6.1 Recursos humanos

- Programador

Nombre:

Ignacio Rodríguez

Descripción del cargo y tareas a realizar:

El programador es quien se encarga de que el sistema funcione correctamente y sin fallas, además de que su interfaz sea amigable con el usuario y fácil de utilizar. Tiene la responsabilidad de terminar el trabajo en fecha.

- ## Documentador

Nombre:

Agustina Martínez

Descripción del cargo y tareas a realizar:

El documentador es quien se encarga de documentar todo lo relacionado con el proyecto. Tiene la responsabilidad de terminar el trabajo en fecha y de realizarlo con prolijidad.

- ## Tester

Nombre:

Personal a contratar capacitado en el área.

Descripción del cargo y tareas a realizar:

El tester es quien se encarga de probar el programa y tratar de encontrar sus fallas, para que luego el programador pueda solucionarlas. También brinda posibles soluciones al problema.

- ## Vendedores

Nombre:

Personal a contratar capacitado en el área.

Descripción del cargo y tareas a realizar:

El vendedor es quien se encarga de promocionar los productos de la empresa de la mejor forma, asegurando que los mismos serán útiles y efectivos para el cliente.

- ## Administrador

Nombre:

Personal a contratar capacitado en el área.

Descripción del cargo y tareas a realizar:

El administrador se encarga de organizar todo lo referido a las tareas administrativas de la empresa.

2.6.2 Recursos financieros

- Capital para la adquisición de los recursos materiales y tecnológicos, y la mantención de los recursos humanos. Este capital provendrá de los socios de la empresa.

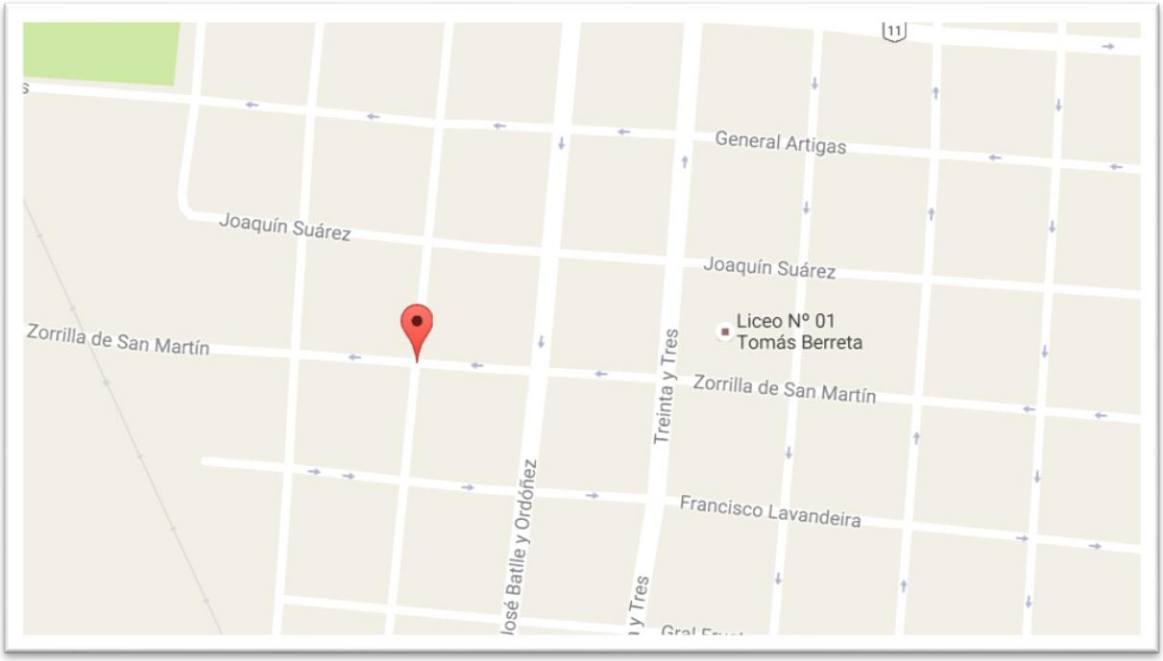
2.6.3 Recursos materiales

- Cable UTP cat5e
- Switch
- Computadoras con las siguientes especificaciones:
  - Procesador: Intel Dual Core g3260 3,3ghz
  - Mother GA-H81M-S1 Gigabyte con USB 3.0
  - Memoria 4Gb DDR3
  - Disco sata III 500gb
- Monitor
- Teclado USB
- Conectores RJ45
- Ductos
- Cajas de registro
- Mouse USB
- Raspberry Pi 2
- Oficina para la localización de la empresa

2.6.4 Recursos tecnológicos

- Computadoras
- Software necesario para programar
- Licencias de Windows 7
- Licencias de Visual Basic 2013

2.7 Localización y entorno de la empresa



Nuestra empresa tiene sus oficinas en un punto estratégico de la ciudad de Canelones, a pocas cuadras de la calle principal.

Consideramos a ésta como una buena ubicación debido a que es un punto transitado de la ciudad, debido a esto la empresa tiene más posibilidades de conseguir clientes.

El entorno de la empresa es favorable, ya que a sus alrededores se pueden encontrar muy pocos competidores. Además, los proveedores tendrán fácil acceso al local de la empresa debido a su ubicación.

También nos encontramos cerca del cliente, lo cual es una gran ventaja.

2.8 Forma jurídica de la empresa: S.R.L.

MAITS optó por la forma jurídica SRL ya que es un tipo de sociedad conformada por socios, en la cual la responsabilidad está limitada al capital aportado por los mismos, y por lo tanto, en el caso de que se contraigan deudas, no se responde con su patrimonio personal.

2.8.1 Ventajas de las empresas S.R.L.

- Protege a los socios en caso de una demanda a la sociedad y a los activos de la empresa cuando haya una demanda en contra de cualquier socio.
- Una SRL requiere de un mínimo de 2 socios y un máximo de 50.
- Los socios pueden designar a otra entidad para que administre la compañía en su nombre.
- La SRL goza de duración perpetua, a menos que el acta constitutiva indique lo contrario.

2.8.2 Desventajas de las empresas S.R.L.

- La admisión de nuevos socios o la venta de las participaciones requiere del consentimiento de los otros.
- El capital empresarial no puede aumentar, a menos que sea efectivamente pagado, lo que limita la expansión de la empresa.

3. Proyecto

3.1 Descripción del proyecto

El objetivo de nuestro proyecto es desarrollar una herramienta que permita la gestión de bedelía. Dicha herramienta posee una interfaz amigable, intuitiva y fácil de entender/utilizar en el momento requerido.

Mediante el ingreso al sistema de grupos, profesores, salones disponibles y asignaturas, el sistema facilitará la distribución de horarios, notificando al usuario en caso de que haya fallas (ej: superposición de horarios).

3.2 Fundamentación del proyecto

Mediante la creación de ésta herramienta, MAITS Software pretende facilitar el trabajo de los funcionarios. Si la misma es utilizada se ahorrará tiempo, el cual podría ser utilizado en la realización de otras tareas.

En base a datos recopilados por la empresa, en el antiguo método de asignación de horarios surgían muchos problemas, los cuales se van a reducir significativamente si se utiliza nuestro software.

Cabe destacar que MINERVA no solo será útil para un centro educativo, sino que también puede ser adaptado para ser utilizado en más centros educativos.

3.3 Análisis FODA

3.3.1 Fortalezas

- Personal con ganas de brindar todo su potencial.
- Ofrecimiento de servicio técnico.
- Empresa con afán de progresar en el mercado.

3.3.2 Oportunidades

- Bajo número de competidores en la región.
- Avance de la tecnología en el mundo.
- Aumento del uso de la informática en los centros educativos.

3.3.3 Debilidades

- Poca experiencia de los emprendedores.
- Al ser una empresa nueva, será poco conocida en el mercado.
- No existe personal capacitado en las áreas publicidad y administración.
- Escasos recursos financieros.

3.3.4 Amenazas

- Posible aparición de competencias.
- No está asegurado que todas las instituciones quieran innovar el uso de tecnología mediante el uso del sistema.
-

3.4 Estudio de Factibilidad

3.4.1 Factibilidad técnica

Los equipos utilizados por MATIS Software poseen las características necesarias para el correcto funcionamiento del sistema, por lo cual no se dificultará al momento de utilizar el mismo.

No se requieren tecnologías adicionales a las ya descriptas anteriormente.

3.4.2 Factibilidad económica

El sistema es factible económicamente ya que consume menos tiempo del que consume la actual manera de designación de horarios (a lápiz y papel). *

Por lo tanto, además de disminuir el trabajo de los empleados, les brindará más tiempo libre para destinar a otras tareas, con lo cual la UTU aumentará su productividad.

** Cálculo basado en las prestaciones del programa y en estudios previos a la creación del mismo.*

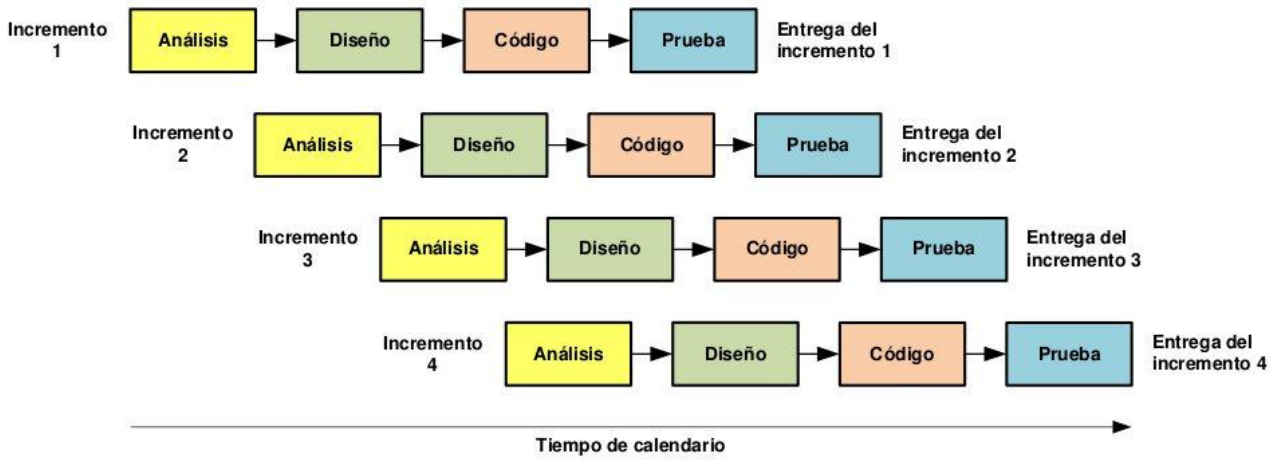
3.4.3 Factibilidad operativa

Los recursos humanos de la empresa están capacitados para la utilización del sistema. Si se necesita capacitar a otros usuarios, se hará previamente a la instalación para que no se presenten inconvenientes.

3.4.4 Factibilidad legal

El proyecto no infringe ninguna ley o norma establecida, por lo tanto es factible legalmente.

3.5 Ciclo de vida del Software: Incremental



Bajo este modelo se entrega software por partes funcionales más pequeñas pero reutilizables, llamadas incrementos. En general cada incremento se construye sobre aquel que ya fue entregado.

3.5.1 Características del modelo incremental

- Cada incremento agrega funcionalidad adicional o mejorada sobre el sistema.
- A partir de la evaluación se planea el siguiente incremento y así sucesivamente.
- En lugar de entrega del sistema en una sola entrega, el desarrollo y la entrega están fracturados bajo incrementos, con cada incremento se entrega parte de la funcionalidad requerida
- Se evitan proyectos largos y se entrega “algo de valor” a los usuarios con cierta frecuencia.

3.5.2 Ventajas del modelo incremental

- Los clientes no tienen que esperar hasta que el sistema se entregue completamente para comenzar a hacer uso de él.
- Los clientes pueden usar los incrementos iniciales como prototipo para precisar los requerimientos posteriores del sistema.
- Minimización del riesgo de falla en el proyecto porque los errores se van corrigiendo progresivamente.
- Construir un sistema pequeño es siempre menos riesgoso que construir un sistema grande.
- Si un error importante es realizado, sólo la última iteración necesita ser descartada.

3.6 Aseguramiento de calidad

3.6.1 Verificación

- ¿Estamos construyendo el producto correctamente?

El papel de la verificación comprende comprobar que el software está de acuerdo con su especificación. Se comprueba que el sistema cumple los requerimientos funcionales y no funcionales que se le han especificado.

3.6.2 Validación

- ¿Estamos construyendo el producto correcto?

La validación es un proceso más general. Se debe asegurar que el software cumple las expectativas del cliente. Va más allá de comprobar si el sistema está acorde con su especificación, para probar que el software hace lo que el usuario espera a diferencia de lo que se ha especificado.

4. Especificación de Requerimientos

4.1 Introducción

4.1.1 Propósito del documento

Este documento contiene la Especificación de Requerimientos del Sistema de Gestión de Bedelía Minerva. Se describen en éste los requerimientos funcionales y no funcionales, casos de uso, funciones, restricciones y otros factores necesarios para una definición correcta del sistema anteriormente nombrado.

4.1.2 Propósito del sistema

Ofrecer una herramienta que facilite la asignación de horarios docentes al inicio del curso en la Escuela Técnica Canelones.

4.1.3 Situación actual

Actualmente la asignación de horarios se realiza manualmente. Para ello se toma en cuenta la antigüedad de los docentes y al recibir la boleta docente se pide a los mismos una aspiración horaria, dado que es común que los docentes también trabajen fuera de la institución. Esta boleta contiene el área, la asignatura, el nivel, la orientación, la cantidad de horas y el turno.

A fin de comprobar que no haya superposiciones de horarios de docentes, se colocan los horarios de todos los cursos y se enfrentan.

Cabe destacar que éstas actividades se realizan a lápiz y papel, y consumen mucho tiempo al momento de llevarlas a cabo.

4.1.4 Definiciones, acrónimos y abreviaturas

- **Hardware:** Conjunto de elementos materiales que constituyen el soporte físico de un ordenador.
- **Software**: Término genérico que se aplica a los componentes no físicos de un sistema informático,como p. ej. los programas,sistemas operativos,etc.,que permiten a este ejecutar sus tareas.
- **ESR:** Especificación de Requerimientos
- **IDE:** Interfaz de desarrollo integrado, ambiente de desarrollo de un lenguaje de programación.
- **Link o enlace:** Un enlace o link es texto o imágenes en un sitio web que un usuario puede hacer click para tener acceso o conectar con otro documento.
- **Grilla o grid:** Cuadrícula para presentar datos en forma de tabla.
- **Tipografía o fuente:** Tipo de letra.
- **RF:** Requerimiento funcional.
- **Proyecto:** conjunto de las actividades que desarrolla una persona o una entidad para alcanzar un determinado objetivo. Estas actividades se encuentran interrelacionadas y se desarrollan de manera coordinada.
- **Raspberry Pi:** Se trata de una placa de bajo coste diseñada en Reino Unido por la Fundación Raspberry Pi. Fabricada para estimular la enseñanza de la ciencias de la computación en las escuelas o en casa. Su principal objetivo es ejecutar una distribución linux o RISC OS.
- **Servidor:** La definición de servidor nos dice que es un equipo que se encarga de dar algún tipo de servicio a otros PCs que se denominan clientes. Se trata de máquinas que normalmente va a estar funcionando 24 horas al día y los 365 días del año. Ejemplos de trabajos que realizan son páginas Web, base de datos, servidores de ficheros, servidores de impresión, etc.

4.1.5 Referencias

- proyecto3bg2016v2.0.docx.pdf - Letra del proyecto (2016)
- ESREHankieResumido.pdf - Ejemplo ESR
- ieee830.pdf - Modelo ESR

4.2 Alcance y limitaciones

4.2.1 Alcance

El sistema a desarrollarse proveerá una herramienta para la facilitación del trabajo de bedelía de la Escuela Técnica Canelones a la hora de asignar los horarios de cada grupo, así como ofrecerá la posibilidad de cambio de horarios en caso de imprevistos.

También existe la posibilidad de adaptar el sistema para su utilización en cualquier institución.

4.2.2 Limitaciones:

Software

El software sólo será soportado en versiones de Windows superiores o iguales a Windows 7.

Además se limitó la herramienta de desarrollo del mismo al lenguaje de programación VB.NET , utilizando el IDE Visual Studio 2013.

Hardware

Para la base de datos se utilizará un Raspberry Pi 2 con las siguientes características:

- Procesador: 900MHz quad-core ARM Cortex-A7 CPU
- Memoria: 1GB
- 4 puertos USB

- 40 pines GPIO
- Puerto HDMI
- Puerto Ethernet
- microSD

Se utilizarán cuatro equipos con las siguientes características para el uso del sistema:

- Intel Dual Core g3260 3,3ghz
- Mother GA-H81M-S1 Gigabyte con USB 3.0
- Memoria 4Gb DDR3
- Disco sata III 500gb

La empresa no dispone de estos equipos actualmente pero se posee la autorización para la compra de los mismos.

El costo estimado es de **U$S2015** por las cuatro PCs y el Raspberry Pi 2.

Tiempo

El sistema Minerva deberá estar funcionando antes del día **22/10/2016.**

Costo

El sistema tendrá un costo aproximado de U$S 29.300.

4.3 Descripción general

4.3.1 Funciones del producto

0. Alta, baja y modificación de usuarios del sistema
1. Manejo de docentes al sistema
2. Manejo de grupos
3. Manejo de salones
4. Manejo de turnos
5. Manejo de orientaciones

6. Manejo de asignaturas
7. Disposición y modificación de horarios
8. Consulta de horarios por grupo, turno y grado

4.3.2 Actores del sistema

Los usuarios ingresados al sistema se categorizarán bajo roles, cada uno de los cuales posee sus permisos y restricciones correspondientes.

Los niveles de acceso al sistema serán:

Consultante:

- Consulta de horarios y grupos, los cuales detallan la materia y docente
- Entrar como invitado

Adscripto:

- Login
- Registro
- Consulta de grupos
- Consulta de salones
- Asignación de salones

Funcionario:

- Login
- Registro
- Manejo de docentes
- Manejo de grupos
- Manejo de salones
- Manejo de asignaturas tomadas
- Disposición y modificación de horarios
- Consulta de horarios por grupo, los cuales detallan la materia y docente

Administrador:

- Login
- Alta, baja y modificación de usuarios del sistema
- Manejo de docentes
- Manejo de grupos
- Manejo de salones
- Manejo de asignaturas tomadas
- Disposición y modificación de horarios
- Consulta de horarios por grupo, los cuales detallan la materia y docente
- Aprobación de usuario

4.4 Requerimientos

4.4.1 Requerimientos funcionales

RF01 - Consulta de horarios por grupo

El sistema deberá permitir consultar los horarios asignados a cada grupo tanto como por día como por semana. En la consulta se encontrarán detalladas las materias y docentes.

RF02 - Alta de docentes

El sistema deberá poder dar de alta a un docente al ingresar los datos que son requeridos por el programa (CI, Nombre, Apellido, Cargo)

RF03 - Baja o modificación de docentes

El sistema deberá poder dar de baja a un docente siempre y cuando el mismo no posea horas asignadas. También permitirá la modificación de los mismos.

RF4 - Alta de grupos

El sistema deberá poder dar de alta a un grupo ingresando los siguientes datos:

- Carga horaria por materia
- Tipo de curso
- Grado (1, 2 o 3)
- ¿Posee algún alumno con alguna discapacidad? (V o F)

RF05 - Baja o modificación de de grupos

El sistema deberá poder dar de baja a un grupo siempre y cuando no se encuentre ningún alumno anotado para el mismo. También permitirá la modificación de los mismos.

RF06 - Alta de salones

El sistema deberá poder dar de alta un salón ingresando los siguientes datos:

- ID Salón
- Planta
- Comentarios del salón

La asignación de salones a grupos es opcional.

RF07 - Baja o modificación de salones

El sistema deberá poder dar de baja un salón siempre y cuando el mismo no se encuentre ocupado en el momento. También permitirá la modificación de los mismos.
Los salones no deben estar asignados a ningún grupo para poder ser eliminados.
La misma asignación puede ser modificada.

RF08 - Disposición de horarios

Una vez agregados los datos del grupo, el sistema repartirá las materias automáticamente.

RF09 - Modificación de horarios

El administrador o director serán capaces que modificar los horarios generados por el programa en RF08, que serán reflejados en una grilla.

RF10 - Alta de usuarios del sistema

El sistema deberá poder dar de alta a un usuario ingresando los siguientes datos:

- Nombre de usuario
- Contraseña

Dicho usuario deberá ser aprobado por el administrador una vez creado.
Cabe destacar que el primer usuario en registrarse, será automáticamente el administrador.

RF11 - Baja de usuarios del sistema

El sistema deberá poder dar de baja a un usuario siempre y cuando el usuario que desea realizarlo posea los permisos pertinentes.

RF12 - Modificación de usuarios del sistema

El sistema deberá permitir la modificación de usuarios registrados al sistema siempre y cuando el usuario que desea realizarlo posea los permisos pertinentes.
El usuario puede ser aprobado o desaprobado en el momento de la modificación.

4.4.2 Requerimientos no funcionales

Hardware

Para el servidor (Raspberry Pi 2)

- Procesador: 900MHz quad-core ARM Cortex-A7 CPU
- Memoria: 1GB
- 4 puertos USB
- 40 pines GPIO
- Puerto HDMI
- Puerto Ethernet
- microSD slot

Para terminales

- Procesador: Intel Dual Core g3260 3,3ghz
- Memoria: 4GB DDR3
- Disco sata III 500GB
- Mother GA-H81M-S1 Gigabyte con USB 3.0

Software

El sistema debe ser desarrollado con el lenguaje de programación .NET, siguiendo los estándares de nomenclatura de elementos disponibles en: https://msdn.microsoft.com/en-us/library/aa263493(v=vs.60).aspx. La interface

de desarrollo (IDE) será ser Visual Studio 2013, la cual es de pago pero su versión de prueba se encuentra disponible en el sitio: https://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx

El sistema de gestión de base de datos será MySQL, la cual necesitará el conector .NET para conectarse con Visual Studio 2013 y se hallará instalado en un servidor remoto Linux. Estos pueden encontrarse de forma gratuita en los siguientes links:

- MySQL Installer https://dev.mysql.com/downloads/installer/
- Ubuntu 16.04 LTS http://www.ubuntu.com/download/desktop/
- Conector .NET https://dev.mysql.com/downloads/connector/net/

El sistema necesitará tener instalado .Net Framework 4.5 para funcionar correctamente.

- Net Framework 4.5 https://www.microsoft.com/en-us/download/details.aspx?id=30653

Nuestro sistema podrá ejecutarse en los siguientes sistemas operativos:

- Windows 7
- Windows 8/8.1
- Windows 10

Seguridad y control de acceso:

Los usuarios ingresarán al sistema a través de un usuario y contraseña que estará guardado en la base de datos. Cada usuario será clasificado en distintos tipos: Consultas, Administrador o Director. Luego de ingresado al sistema, el mismo determinará en base al tipo de cuenta asignada a que opciones del mismo el usuario podrá acceder.

Integración con otros sistemas:

El sistema poseerá su propia base de datos y no estará integrado a ningún sistema existente.

Ya existe una base de datos, pero MAITS no posee acceso a la misma.

Tipografías:

Las tipografías o fuentes utilizadas en el sistema se encuentran disponibles en la carpeta del mismo

Interfaz con el usuario:

No existen requerimientos previos respecto a la interfaz de usuario, solamente que ésta sea amigable y fácil de utilizar.

Ayuda online:

Minerva no contará con ayuda online

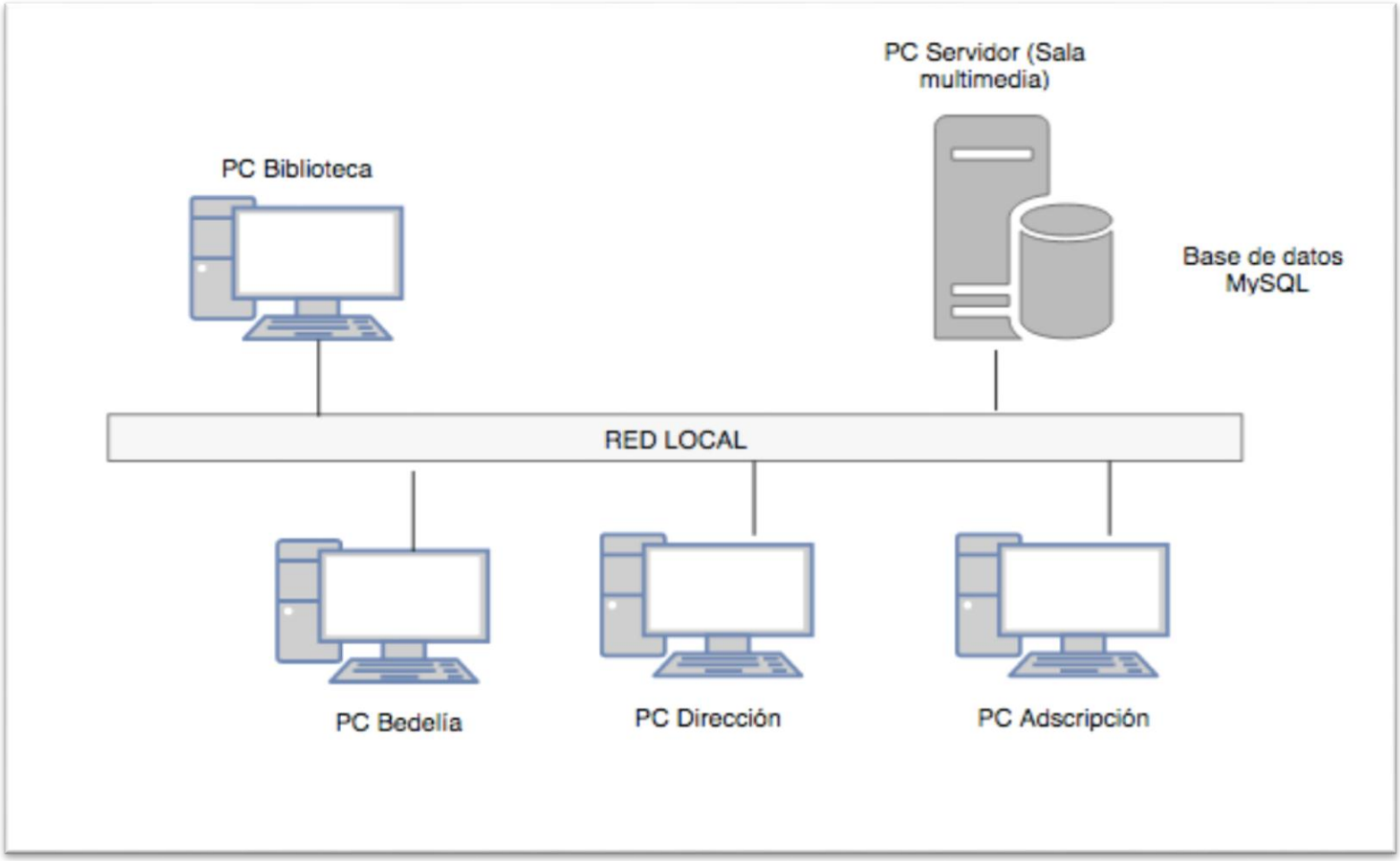
Interfaz online:

Minerva no contará con interfaz online.

Requerimientos internacionales, legales y otros:

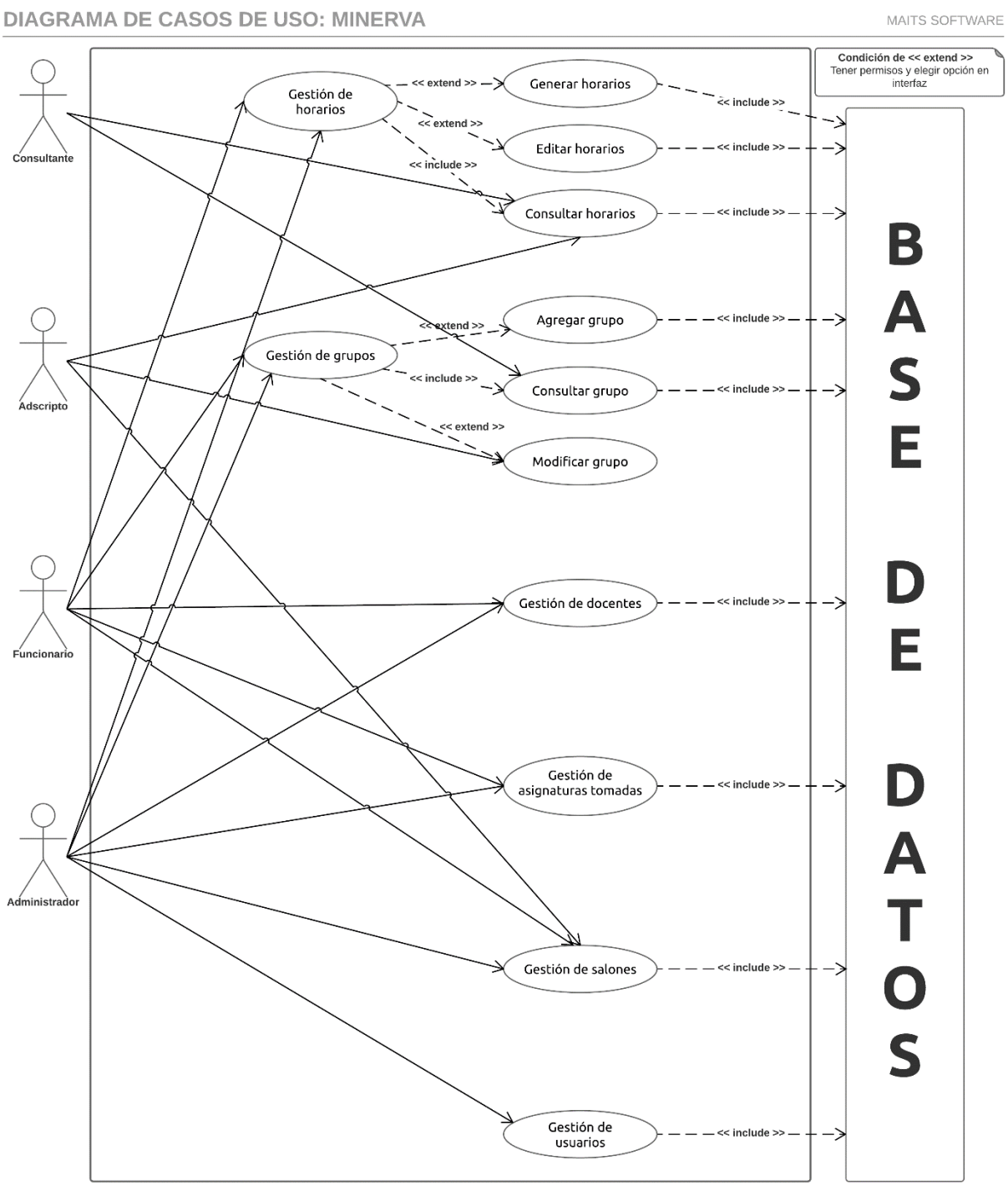
No especificados hasta el momento.

4.5 Diagrama de despliegue



4.6 Casos de uso

4.6.1 Diagrama de casos de uso



4.6.2 Casos de uso especificados

Acceso al sistema

Nombre	Acceso al sistema.
Actores	• Administrador • Funcionario • Consultante • Adscripto • Base de datos MySQL
Descripción	Luego de abrir el programa, se permitirá el uso del sistema.
Precondiciones	• Abrir el programa.
Flujo principal	1. Usuario: ejecuta el programa 2. Sistema: verifica la conexión con la base de datos.
Flujo alternativo	*Error en la conexión con la base de datos:* 2.1 Sistema: notifica al usuario del error.
Poscondiciones	• El programa se podrá utilizar.

Login (no contemplado por UML)

Nombre	Login
Actores	• Administrador • Funcionario • Consultante • Adscripto • Base de datos MySQL
Descripción	El usuario se loguea en el sistema.
Precondiciones	• Programa abierto y conectado a la base de datos.
Flujo principal	1. Usuario: escoge la opción “ingresar” en la ventana principal. 2. Usuario: ingresa sus datos. 3. Sistema: verifica los datos ingresados. 4. Usuario: ingresa al programa.
Flujo alternativo	*Usuario/contraseña incorrecta:* 3.1 Sistema: Notifica al usuario sobre los datos incorrectos. 3.2 Usuario: Intenta nuevamente *Usuario y contraseña correctos. Cuenta no aprobada por administrador:* 3.1 Sistema: Notifica al usuario sobre dicha situación
Poscondiciones	• El usuario podrá ingresar al sistema.

Registro

Nombre	Registro
Actores	• Administrador • Funcionario • Consultante • Adscripto • Base de datos MySQL
Descripción	Permite el registro de usuarios.
Precondiciones	• Programa abierto y conectado a la base de datos.
Flujo principal	1. Usuario: escoge la opción “registro” en la ventana principal. 2. Usuario: ingresa los datos de registro. 3. Sistema: ejecuta sentencia en base de datos 4. Sistema: notifica al usuario sobre la aprobación pendiente.
Flujo alternativo	*CI de usuario en uso:* 3.1 Sistema: notifica al usuario sobre los datos repetidos 3.1 Usuario: intenta nuevamente *Primer usuario en registrarse:* 4.1 Sistema: notifica al usuario sobre dicha situación 4.2 Sistema: convierte al usuario en administrador.
Poscondiciones	• El usuario registrado podrá administrar el sistema.

Ingreso como invitado (no contemplado por UML)

Nombre	Ingreso como invitado
Actores	• Administrador • Funcionario • Consultante • Adscripto • Base de datos MySQL
Descripción	Ingreso al sistema como invitado (sólo permite consultar)
Precondiciones	• Programa abierto y conectado a la base de datos.
Flujo principal	1. Usuario: escoge la opción “ingresar como invitado” 2. Usuario: ingresa al programa
Flujo alternativo	No existe.
Poscondiciones	• El usuario podrá consultar horarios.

Edición del servidor

Nombre	Edición del servidor
Actores	• Administrador • Funcionario • Consultante • Adscripto • Base de datos MySQL
Descripción	Luego de presionar un botón, el usuario podrá cambiar los datos del servidor.
Precondiciones	• Programa abierto.
Flujo principal	1. Usuario: clickea botón en la ventana inicial. 2. Sistema: carga datos actuales de la conexión al servidor. 3. Usuario: ingresa datos. 4. Usuario: prueba la conexión. 5. Sistema: prueba la conexión.
Flujo alternativo	*Datos de conexión incorrectos* 4.1 y 5.1 Sistema: notifica al usuario. Usuario: ingresa datos nuevamente.
Poscondiciones	• El programa será capaz de establecer la conexión con el servidor.

Selección de grupo

Nombre	Selección de grupo
Actores	• Administrador • Funcionario • Consultante • Adscripto • Base de datos MySQL
Descripción	Luego de seleccionar un grupo el programa muestra los horarios del mismo.
Precondiciones	• Programa abierto y conectado a la base de datos.
Flujo principal	6. Usuario: escoge un grupo 7. Sistema: carga calendario y datos de dicho grupo 8. Sistema: muestra los horarios
Flujo alternativo	*Horarios sin asignar "x" día* 3.1 Sistema: muestra un mensaje alertando la situación.
Poscondiciones	• El usuario podrá visualizar los horarios.

Cerrar sesión

Nombre	Cerrar sesión

Actores	• Administrador • Funcionario • Consultante • Adscripto • Base de datos MySQL
Descripción	Cerrar la sesión del sistema.
Precondiciones	• Programa abierto y conectado a la base de datos.
Flujo principal	1. Usuario: clickea la opción “cerrar sesión” 2. Sistema: destruye la ventana de horarios y vuelve a la ventana inicial donde el usuario puede loguearse, registrarse o entrar como invitado.
Flujo alternativo	No existe.
Poscondiciones	• El programa se cerrará.

Ingreso a la ventana de administración

Nombre	Ingreso a la ventana de administración
Actores	• Administrador • Funcionario • Adscripto • Base de datos MySQL
Descripción	El usuario podrá administrar el sistema (salones, grupos, docentes y sus asignaturas, horarios, usuarios(sólo administrador))
Precondiciones	• Programa abierto y conectado a la base de datos. • Usuario logueado en el sistema.
Flujo principal	1. Usuario clickea la opción “administrar” 2. Sistema: abre la ventana de administración, con pestaña “salones” abierta por defecto.
Flujo alternativo	No existe.
Poscondiciones	Se abrirá la ventana de administración y el usuario podrá manejarla.

Administración de salones

Nombre	Administración de salones
Actores	• Administrador • Funcionario • Adscripto • Base de datos MySQL
Descripción	El usuario puede agregar, consultar, modificar y eliminar salones. También puede asignarlos a grupos.
Precondiciones	• Programa abierto y conectado a la base de datos.

	• Usuario logueado en el sistema.
Flujo principal	1. Usuario: clickea la opción “salones” en la barra de botones. 2. Sistema: carga la lista de salones
Flujo alternativo	No existe.
Poscondiciones	El usuario podrá manejar la ventana de salones.

Agregar salón

Nombre	Agregar salón
Actores	• Administrador • Funcionario • Base de datos MySQL
Descripción	El usuario podrá agregar salones.
Precondiciones	• Programa abierto y conectado a la base de datos. • Usuario logueado en el sistema. • Haber clickeado el botón “nuevo salón”
Flujo principal	1. Usuario: ingresa datos del salón 2. Sistema: verifica y guarda en la base de datos 3. Sistema: recarga la lista de salones 4. Sistema: permite al usuario previsualizar el salón
Flujo alternativo	*Existencia de un salón con la misma ID al que se intenta ingresar:* 2.1 Sistema: notifica al usuario. 2.2 Usuario: vuelve a ingresar los datos.
Poscondiciones	Se agrega el salón a la base de datos.

Previsualizar salón

Nombre	Previsualizar salón
Actores	• Administrador • Consultante • Funcionario • Adscripto • Base de datos MySQL
Descripción	El usuario podrá previsualizar los salones ya ingresados.
Precondiciones	• Programa abierto y conectado a la base de datos.
Flujo principal	1. Usuario: selecciona un grupo de la lista de grupos. 2. Sistema: carga los datos del
Flujo alternativo	No existe.

Poscondiciones	Se previsualizará el salón.

Asignar salón

Nombre	Asignar salón a grupo
Actores	• Administrador • Funcionario • Adscripto • Base de datos MySQL
Descripción	Permite asignar un salón a un grupo previamente ingresado.
Precondiciones	• Programa abierto y conectado a la base de datos. • Usuario logueado en el sistema. • Usuario modificando o agregando un salón
Flujo principal	1. Usuario: selecciona un grupo, dependiendo del turno. 2. Sistema: verifica y escribe en la base de datos.
Flujo alternativo	No existe.
Poscondiciones	El salón será asignado al grupo.

Editar salón

Nombre	Editar salón
Actores	• Administrador • Funcionario • Base de datos MySQL
Descripción	Permite al usuario editar los datos de un salón
Precondiciones	• Programa abierto y conectado a la base de datos. • Usuario logueado en el sistema.
Flujo principal	1. Usuario: clickea la opción “editar”, que aparece al costado de cada salón. 2. Sistema: carga los datos del salón elegido. 3. Usuario: modifica los datos del salón. 4. Sistema: verifica y escribe en la base de datos.
Flujo alternativo	Usuario cancela la edición: 3.1 Usuario: clickea “cancelar edición” 3.2 Sistema: la interfaz vuelve al modo de agregar salón.
Poscondiciones	Se editará el salón.

Eliminar salón

Nombre	Eliminar salón

Actores	• Administrador • Funcionario • Base de datos MySQL
Descripción	Permite al usuario eliminar salones.
Precondiciones	• Programa abierto y conectado a la base de datos. • Usuario logueado en el sistema.
Flujo principal	1. Usuario: clickea la opción “eliminar” que se sitúa a la derecha de cada salón 2. Sistema: pide confirmación al usuario 3. Sistema: elimina el salón de la base de datos 4. Sistema: carga los salones nuevamente
Flujo alternativo	*El usuario no desea borrar el salón:* 2.1 Usuario: presiona cancelar al momento de que se le pide confirmación. Salón ya asignado a un grupo: 3.1 Sistema: notifica al usuario sobre dicho problema e impide que el salón sea borrado.
Poscondiciones	Se eliminará el salón seleccionado.

Administración de grupos

Nombre	Administración de grupos
Actores	• Administrador • Funcionario • Base de datos MySQL
Descripción	Permite al usuario administrar los grupos (eliminar, agregar y previsualizar)
Precondiciones	• Programa abierto y conectado a la base de datos. • Usuario logueado en el sistema.
Flujo principal	1. Usuario clickea la opción “grupos” en la barra de botones. 2. Sistema: carga la lista de grupos.
Flujo alternativo	No existe.
Poscondiciones	Los grupos serán previsualizados.

Editar grupos

Nombre	Editar grupos
Actores	• Administrador • Funcionario • Base de datos MySQL
Descripción	Permite al usuario editar los datos de los grupos

Precondiciones	• Programa abierto y conectado a la base de datos. • Usuario logueado en el sistema.
Flujo principal	1. Usuario: clickea la opción “editar”, que aparece al costado de cada salón. 2. Sistema: carga los datos del grupo elegido. 3. Usuario: modifica los datos del grupo. 4. Sistema: verifica y escribe en la base de datos.
Flujo alternativo	Usuario cancela la edición: 3.1 Usuario: clickea “cancelar edición” 3.2 Sistema: la interfaz vuelve al modo de agregar grupo.
Poscondiciones	Se editará el grupo.

Agregar grupo

Nombre	Agregar grupo
Actores	• Administrador • Funcionario • Base de datos MySQL
Descripción	Permite al usuario agregar grupos al sistema.
Precondiciones	• Programa abierto y conectado a la base de datos. • Usuario logueado en el sistema. • Haber clickeado el botón “nuevo grupo”
Flujo principal	1. Usuario: ingresa datos. 2. Sistema: verifica y escribe en la base de datos. 3. Sistema: recarga la lista de grupos.
Flujo alternativo	*Ya existe un grupo con el mismo grado e ID:* 2.1 Sistema: notifica al usuario 2.2 Usuario: vuelve a ingresar los datos
Poscondiciones	El grupo será agregado al sistema.

Previsualizar grupo

Nombre	Previsualizar grupo
Actores	• Administrador • Consultante • Funcionario • Adscripto • Base de datos MySQL
Descripción	El usuario podrá previsualizar grupos.
Precondiciones	• Programa abierto y conectado a la base de datos.
Flujo principal	1. Usuario: clickea un grupo de la lista. 2. Sistema: carga datos del grupo

Flujo alternativo	No existe.
Poscondiciones	Se visualizarán los datos del grupo.

Eliminar grupo

Nombre	Eliminar grupo
Actores	• Administrador • Funcionario • Base de datos MySQL
Descripción	El usuario podrá eliminar grupos.
Precondiciones	• Programa abierto y conectado a la base de datos. • Usuario logueado en el sistema.
Flujo principal	1. Usuario: clickea la opción “eliminar” situada a la derecha del grupo deseado. 2. Sistema: pide confirmación al usuario. 3. Sistema: elimina el grupo de la base de datos. 4. Sistema: carga los datos.
Flujo alternativo	*El usuario no desea borrar el grupo:* 2.1 Usuario: presiona cancelar al momento de la confirmación *El grupo tiene docentes asignados:* 3.1 Sistema: notifica al usuario sobre dicho problema e impide que el grupo se borre.
Poscondiciones	El grupo será eliminado.

Administración de docentes

Nombre	Administración de docentes.
Actores	• Administrador • Funcionario • Base de datos MySQL
Descripción	El usuario puede agregar, consultar, modificar y eliminar docentes. También puede asignar, consultar y eliminar asignaturas que fueron asignadas.
Precondiciones	• Programa abierto y conectado a la base de datos. • Usuario logueado en el sistema.
Flujo principal	1. Usuario: ingresa datos. 2. Sistema: verifica y escribe en la base de datos. 3. Sistema: recarga la lista de docentes 4. Usuario: modifica las asignaturas del docente. *(Ver caso de uso: Editar Asignaturas)*
Flujo alternativo	*Ya existe un docente con la misma CI* 2.1 Sistema: notifica al usuario 2.2 Usuario: vuelve a ingresar los datos

Poscondiciones	El usuario podrá administrar los datos de los docentes.

Agregar docente

Nombre	Agregar docente
Actores	• Administrador • Funcionario • Base de datos MySQL
Descripción	El usuario podrá agregar docentes.
Precondiciones	• Programa abierto y conectado a la base de datos. • Usuario logueado en el sistema. • Haber clickeado el botón “nuevo docente”
Flujo principal	1. Usuario: ingresa datos 2. Sistema: verifica y escribe en la base de datos 3. Sistema: recarga la lista de docentes 4. Usuario: modifica las asignaturas del docente. *(Ver caso de uso: Editar Asignaturas)*
Flujo alternativo	*Ya existe un docente con la misma CI* 2.1 Sistema: notifica al usuario 2.2 Usuario: vuelve a ingresar datos
Poscondiciones	El docente será agregado.

Previsualizar docente y sus asignaturas

Nombre	Previsualizar docente y sus asignaturas
Actores	• Administrador • Funcionario • Base de datos MySQL • Consultante
Descripción	Se podrán previsualizar datos de los docentes y sus asignaturas.
Precondiciones	• Programa abierto y conectado a la base de datos.
Flujo principal	1. Usuario: selecciona al docente de la lista de grupos. 2. Sistema: carga datos del docente, incluyendo asignaturas tomadas por él.
Flujo alternativo	No existe.
Poscondiciones	El usuario previsualizará los datos.

Editar datos del docente

Nombre	Editar datos del docente

Actores	• Administrador • Funcionario • Base de datos MySQL
Descripción	El usuario podrá editar los datos del docente.
Precondiciones	• Programa abierto y conectado a la base de datos. • Usuario logueado en el sistema.
Flujo principal	1. Usuario clickea en “editar”, elige “datos del docente” 2. Sistema: carga datos y asignaturas del profesor. 3. Usuario: modifica datos 4. Sistema: actualiza los datos en la base de datos.
Flujo alternativo	No existe
Poscondiciones	El docente tendrá nuevos datos.

Editar asignaturas del docente

Nombre	Editar asignaturas del docente
Actores	• Administrador • Funcionario • Base de datos MySQL
Descripción	El usuario podrá editar las asignaturas que fueron asignadas al docente, y/o asignar nuevas.
Precondiciones	• Programa abierto y conectado a la base de datos. • Usuario logueado en el sistema.
Flujo principal	1. Usuario: clickea en “editar”, elige “asignaturas del docente” 2. Sistema: carga datos y asignaturas del profesor 3. Usuario: Modifica datos 4. Sistema: actualiza los datos en la base de datos.
Flujo alternativo	*La materia que se le está intentando asignar a ese profesor en ese grupo, fue previamente asignada a otro:* 4.1 Sistema: notifica al usuario acerca del problema.
Poscondiciones	Las asignaturas del docente serán editadas.

Eliminar asignaturas del docente

Nombre	Eliminar asignaturas del docente
Actores	• Administrador • Funcionario • Base de datos MySQL

Descripción	El usuario podrá eliminar las asignaturas que fueron asignadas al docente.
Precondiciones	• Programa abierto y conectado a la base de datos. • Usuario logueado en el sistema.
Flujo principal	1. Usuario: clickea en “editar”, elige “asignaturas del docente” 2. Sistema: carga datos y asignaturas del profesor. Usuario: selecciona asignatura de la lista de asignaturas. 3. Usuario: clickea la opción “eliminar” (representada por un guión rojo) 4. Sistema: actualiza los datos en la base de datos.
Flujo alternativo	No existe.
Poscondiciones	La asignatura será desasignada de ese docente, y podrá ser asignada a otro.

Eliminar docente

Nombre	Eliminar docente
Actores	• Administrador • Funcionario • Base de datos MySQL
Descripción	El usuario podrá eliminar docentes.
Precondiciones	• Programa abierto y conectado a la base de datos. • Usuario logueado en el sistema.
Flujo principal	1. Usuario: clickea la opción “eliminar” que se ubica a la derecha de cada profesor. 2. Sistema: pide confirmación al usuario. 3. Sistema: elimina al profesor de la base de datos. 4. Sistema: actualiza la lista de profesores.
Flujo alternativo	*El usuario no desea borrar al profesor:* 2.1 Usuario: presiona cancelar al momento de la confirmación *El docente tiene asignaturas tomadas:* 3.1 Sistema: notifica al usuario sobre dicho problema e impide que el docente sea eliminado.
Poscondiciones	El docente será eliminado.

Administración de horarios

Nombre	Administración de horarios.
Actores	• Administrador • Funcionario

	• Base de datos MySQL
Descripción	El usuario puede ordenar y consultar horarios.
Precondiciones	• Programa abierto y conectado a la base de datos. • Usuario logueado en el sistema.
Flujo principal	1. Usuario: clickea la opción “horarios” en la barra de botones 2. Sistema: carga la lista de grupos
Flujo alternativo	No existe
Poscondiciones	Se administrarán los horarios a gusto del usuario.

Consultar horarios

Nombre	Consultar horarios
Actores	• Administrador • Funcionario • Adscripto • Base de datos MySQL • Consultante
Descripción	El usuario podrá previsualizar los horarios de los grupos.
Precondiciones	• Programa abierto y conectado a la base de datos.
Flujo principal	1. Usuario: selecciona un grupo de la lista de grupos. 2. Sistema: carga los horarios del calendario
Flujo alternativo	*El grupo no tiene horarios en el calendario:* 2.1 Sistema: crea una lista con asignaturas en base a los datos del curso.
Poscondiciones	Se mostrará la lista de horarios de cada grupo.

Ordenar horarios

Nombre	Ordenar horarios
Actores	• Administrador • Funcionario • Base de datos MySQL
Descripción	El usuario podrá ordenar los horarios.
Precondiciones	• Programa abierto y conectado a la base de datos. • Usuario logueado en el sistema.
Flujo principal	1. Usuario: selecciona un grupo de la lista de grupos. 2. Sistema: carga los horarios del calendario. 3. Usuario: modifica los horarios.

	4. Sistema: guarda los horarios en la base de datos.
Flujo alternativo	No existe.
Poscondiciones	Los horarios se ordenarán a gusto del usuario.

Administración de usuarios

Nombre	Administración de usuarios
Actores	• Administrador • Base de datos MySQL
Descripción	El administrador puede agregar, modificar y eliminar usuarios
Precondiciones	• Programa abierto y conectado a la base de datos. • Usuario logueado en el sistema como administrador.
Flujo principal	1. Administrador: clickea la opción “usuarios” en la barra de botones. 2. Sistema: carga la lista de usuarios
Flujo alternativo	No existe
Poscondiciones	El administrador editará los usuarios.

Agregar usuario

Nombre	Agregar usuario
Actores	• Administrador • Base de datos MySQL
Descripción	El administrador puede agregar usuarios.
Precondiciones	• Programa abierto y conectado a la base de datos. • Usuario logueado en el sistema como administrador.
Flujo principal	1. Administrador: clickea la opción “nuevo usuario” 2. Administrador: ingresa datos. 3. Sistema: guarda en la base de datos.
Flujo alternativo	*El usuario ya existe:* 3.1 Sistema: notifica al administrador e impide que el usuario sea agregado nuevamente.
Poscondiciones	El administrador agregará un nuevo usuario.

Modificar usuario

Nombre	Modificar usuario
Actores	• Administrador

	• Base de datos MySQL
Descripción	El administrador puede modificar usuarios.
Precondiciones	• Programa abierto y conectado a la base de datos. • Usuario logueado en el sistema como administrador.
Flujo principal	1. Administrador: clickea el botón “editar” situado a la derecha del usuario. 2. Sistema: carga datos del usuario: 3. Administrador: ingresa los datos del usuario. 4. Sistema: guarda los datos en la base de datos.
Flujo alternativo	No existe
Poscondiciones	*Contraseña no escrita:* 4.1 Sistema: notifica al usuario.

Ventana inicial

Nombre	Apertura del sistema, ventana Inicial.
Actores	• Administrador • Funcionario • Adscripto • Consultante • Base de datos MySQL
Descripción	El usuario ejecuta Minerva.
Precondiciones	• Programa instalado en el ordenador.
Flujo principal	1. Usuario: Ejecuta el programa. 2. Sistema: Muestra el programa 3. Sistema: Checkea conexion 4. Usuario: Continua su uso regular.
Flujo alternativo	*Error al establecer la conexión con el servidor* 3.1 Sistema: Notifica al usuario *Usuario clickea Ingresar* 4.1 Sistema: Abre ventana de ingreso *Usuario clickea Registrarse* 4.1 Sistema: Abre ventana de registro de usuarios *Usuario clickea Entrar como invitado* 4.1 Sistema: Abre Minerva. *Usuario clickea botón de edición de datos de servidor* 4.1 Sistema: Abre ventana de edición de datos de servidor
Poscondiciones	El usuario podrá cambiar la configuración del servidor (en caso de flujo alternativo). El usuario podrá continuar con el uso normal.

Ventana edición datos del servidor

Nombre	Edición de datos del servidor
Actores	• Administrador • Funcionario • Adscripto • Consultante • Base de datos MySQL
Descripción	El usuario cambia, o verifica los datos de conexión.
Precondiciones	• Clickear opción en ventana inicial.
Flujo principal	1. Sistema: Comprueba conexion 2. Usuario: Ingresa datos 3. Sistema: Comprueba datos ingresados 4. Usuario: Confirma cambios
Flujo alternativo	*Datos de conexión incorrectos* 3.1 Sistema: Notifica al usuario
Poscondiciones	El usuario debería ser capaz de utilizar el programa de forma regular, solo que ésta vez, utilizando otro servidor (o el mismo en su defecto).

Ventana ingreso

Nombre	Ingreso al sistema
Actores	• Administrador • Funcionario • Adscripto • Consultante • Base de datos MySQL
Descripción	El usuario ingresa al sistema.
Precondiciones	• Clickear opcion “Ingresar” en ventana inicial. • Programa abierto y conectado a la base de datos.
Flujo principal	1. Usuario: Ingresa datos de ingreso… 2. Sistema: Comprueba datos 3. Sistema: Abre la ventana principal (Minerva).
Flujo alternativo	*Datos incorrectos* 3.1 Sistema: Notifica al usuario 3.2 Usuario: Vuelve a ingresar datos *Datos correctos, cuenta no aprobada* 3.1 Sistema: Notifica al usuario.
Poscondiciones	El usuario accederá al sistema (Minerva).

Ventana registro

Nombre	Registro de usuario
Actores	• Administrador • Funcionario • Adscripto • Consultante • Base de datos MySQL
Descripción	El usuario se registra en el sistema
Precondiciones	• Clickear opcion “Registrarse” en ventana inicial. • Programa abierto y conectado a la base de datos.
Flujo principal	1. Usuario: Ingresa datos de registro 2. Sistema: comprueba datos
Flujo alternativo	*Primer usuario administrador en el servidor* 3.1 Sistema: Selecciona automáticamente la opción de administrador en tipo de usuario. *Datos correctos, cuenta no aprobada* 3.1 Sistema: Notifica al usuario. *Datos correctos, cuenta aprobada* 3.1 Sistema: Notifica al usuario de que es administrador.
Poscondiciones	El usuario podrá loguearse en el sistema (o no, ver caso de uso de ingreso).

Ventana principal

Cerrar sesión

Nombre	El usuario actual cierra sesión
Actores	• Administrador • Funcionario • Adscripto • Consultante • Base de datos MySQL
Descripción	Cierra el programa (Minerva) y muestra la ventana inicial.
Precondiciones	• Programa abierto y conectado a la base de datos.
Flujo principal	1. Sistema: Mostrar ventana inicial
Flujo alternativo	No posee
Poscondiciones	Se muestra la ventana inicial.

Usuario sin nombre y apellido

Nombre	Ingreso al sistema de usuario sin nombre o apellido
Actores	• Administrador • Funcionario • Adscripto • Base de datos MySQL
Descripción	Al ingresar al sistema, si el usuario no tiene un nombre o apellido, se le pide uno.
Precondiciones	• Programa abierto y conectado a la base de datos. • Usuario sin nombre o apellido
Flujo principal	2. Usuario: Ingresa datos 3. Sistema: Escribe a la DB.
Flujo alternativo	*Faltan datos* 2.1 Sistema: Notifica al usuario 2.2 Usuario: Vuelve a ingresar datos
Poscondiciones	El nombre del usuario será mostrado en pantalla.

Selección de grupo

Nombre	Selección de grupo a visualizar horarios e información.
Actores	• Administrador • Funcionario • Adscripto • Consultante • Base de datos MySQL
Descripción	El usuario selecciona un grupo de la lista. Los datos de dicho grupo (incluyendo horarios), son mostrados en pantalla.
Precondiciones	• Programa abierto y conectado a la base de datos. • Que haya al menos un grupo agregado
Flujo principal	1. Usuario: selecciona grupo 2. Sistema: carga información
Flujo alternativo	No posee
Poscondiciones	Los datos del grupo (y horarios si tiene), serán mostrados en pantalla.

Recargar lista de grupos

Nombre	Recargar la lista de grupos

Actores	• Administrador • Funcionario • Adscripto • Consultante • Base de datos MySQL
Descripción	Recarga la lista de grupos del sistema en la lista de selección.
Precondiciones	• Programa abierto y conectado a la base de datos.
Flujo principal	1. Sistema: recargar lista de grupos teniendo en cuenta si el filtrado está activo. 2. Sistema: recarga información/horarios del grupo seleccionado (si hay uno).
Flujo alternativo	No posee
Poscondiciones	La lista de grupos se actualiza.

Recargar horarios del grupo

Nombre	Recargar los horarios del grupo seleccionado
Actores	• Administrador • Funcionario • Adscripto • Consultante • Base de datos MySQL
Descripción	Recarga los horarios del grupo seleccionado (en la vista)
Precondiciones	• Tener un grupo seleccionado • Programa abierto y conectado a la base de datos.
Flujo principal	1. Sistema: recrea la vista seleccionada con los nuevos datos.
Flujo alternativo	No posee
Poscondiciones	Los horarios se actualizan.

Filtrado de grupo

Nombre	Filtrado de grupo por curso y/o turno.
Actores	• Administrador • Funcionario • Adscripto • Consultante • Base de datos MySQL

Descripción	El usuario selecciona los datos que desea filtrar.
Precondiciones	• Programa abierto y conectado a la base de datos.
Flujo principal	1. Usuario: selecciona datos a filtrar 2. Sistema: recarga la lista de grupos
Flujo alternativo	
Poscondiciones	Los datos del grupo actual (si cumplen los requisitos del filtro): se vuelven a cagar.

Alternar modo de vista

Nombre	Alterna el modo de vista (Semana o Diario)
Actores	• Administrador • Funcionario • Adscripto • Consultante • Base de datos MySQL
Descripción	El usuario selecciona de que forma ver los horarios del grupo.
Precondiciones	• Tener un grupo seleccionado • Programa abierto y conectado a la base de datos.
Flujo principal	1. Usuario: selecciona el modo de vista 2. Sistema: muestra la vista seleccionada
Flujo alternativo	*Mostrar botones adicionales (o ocultarlos)* 2.1 Sistema: Oculta o muestra los botones de la vista semanal (lista)
Poscondiciones	La forma de ver datos cambia.

Guardar grilla de horarios a PDF

Nombre	Guardar a un PDF los horarios.
Actores	• Administrador • Funcionario • Adscripto • Consultante • Base de datos MySQL
Descripción	Guarda a un PDF los horarios de la vista de lista.
Precondiciones	• Tener un grupo seleccionado • Estar en modo: vista de lista • Programa abierto y conectado a la base de datos.

Flujo principal	1. Usuario: Selecciona ubicación a donde guardar el PDF 2. Sistema: guarda el pdf
Flujo alternativo	No posee
Poscondiciones	Continúa el uso normal.

Abrir ventana de administración

Nombre	Abrir ventana de administración
Actores	• Administrador • Funcionario • Adscripto • Base de datos MySQL
Descripción	Abre la ventana en la cual se pueden administrar salones, grupos, horarios, etc (todo depende de los permisos del usuario).
Precondiciones	• Click en opción de interfaz. • Programa abierto y conectado a la base de datos.
Flujo principal	1. Sistema: abre ventana de administración
Flujo alternativo	No posee
Poscondiciones	El usuario trabaja con la ventana de administración.

Ver horarios en ventana externa

Nombre	Ver horarios en una ventana externa.
Actores	• Administrador • Funcionario • Adscripto • Consultante • Base de datos MySQL
Descripción	Abre una ventana externa en la cual se puede elegir el grupo y guardar como PDF sus horarios.
Precondiciones	• Tener un grupo seleccionado • Estar en modo: vista de lista • Programa abierto y conectado a la base de datos.
Flujo principal	1. Sistema: Abre ventana externa y carga el grupo
Flujo alternativo	No posee
Poscondiciones	Usuario se encuentra en la ventana externa, la cual tiene una opción para volver al programa.

Ventana de administración

Apertura

Nombre	Apertura de ventana de administración
Actores	• Administrador • Funcionario • Adscripto • Base de datos MySQL
Descripción	Muestra la ventana de administración y habilita/deshabilita botones en base a sus permisos.
Precondiciones	• Click en opción de interfaz • Programa abierto y conectado a la base de datos.
Flujo principal	1. Sistema: Habilita (muestra) todos los botones
Flujo alternativo	*Usuario adscripto* 1.1 Sistema: Sólo habilita los botones de administración de "Salones" y administración de "Grupos" *Usuario funcionario* 1.1 Sistema: Habilita *casi todos los botones*, sin incluir el de administración de "Usuarios", y "backup a archivo sql".
Poscondiciones	Usuario se encuentra en ventana de administración, en la cual, en base a sus permisos, puede manipular datos.

Selección de opción

Nombre	Selección de opción (**X** a administrar)
Actores	• Administrador • Funcionario • Adscripto • Base de datos MySQL
Descripción	Muestra los controles de lo que el usuario vaya a editar (ej: usuarios)
Precondiciones	• Click en botón de administración de **<X>** • Programa abierto y conectado a la base de datos.
Flujo principal	1. Sistema: Muestra controles de administración de **<X>**
Flujo alternativo	No posee
Poscondiciones	Usuario se encuentra en el formulario de administración de **<X>**. La ventana de administración no se encarga de nada más.

Cierre de la ventana de administración

Nombre	Cierra de ventana de administración
Actores	• Administrador • Funcionario • Adscripto • Base de datos MySQL
Descripción	Al cerrar la ventana de administración, se muestra la ventana principal (Minerva).
Precondiciones	• Cerrar la ventana de administración • Programa abierto y conectado a la base de datos.
Flujo principal	**1.** Sistema: Trae al frente la ventana principal (Minerva).
Flujo alternativo	No posee
Poscondiciones	El usuario continúa con su uso normal.

Administración de salones

Apertura

Nombre	Apertura de la administración de salones.
Actores	• Administrador • Funcionario • Adscripto • Base de datos MySQL
Descripción	Muestra el formulario de administración de salones
Precondiciones	• Click en opción de interfaz • Programa abierto y conectado a la base de datos.
Flujo principal	1. Sistema: Carga la lista de salones 2. Sistema: Prepara interfaz para nuevo salón
Flujo alternativo	*Usuario adscripto* 1.1 Sistema: Al agregarlos deshabilita la edición/eliminación de salones. 1.2 Sistema: Carga por defecto el primer salón 1.3 Sistema: Deshabilita agregación de salones
Poscondiciones	Usuario se encuentra en el formulario de salones, en el cual, en base a sus permisos, puede manipular datos.

Agregar salón

Nombre	Agregar salón

Actores	• Administrador • Funcionario • Base de datos MySQL
Descripción	Agrega el salón a la base de datos
Precondiciones	• Click en opción de interfaz • Programa abierto y conectado a la base de datos.
Flujo principal	1. Usuario: Ingresa datos 2. Sistema: Comprueba datos y escribe en DB 3. Sistema: Carga la lista de salones
Flujo alternativo	*El ID de salón ya existe* 2.1 Notifica al usuario 2.2 El usuario ingresa datos nuevamente
Poscondiciones	El salón es previsualizado

Editar salón

Nombre	Editar salón
Actores	• Administrador • Funcionario • Base de datos MySQL
Descripción	Edita los datos del salón
Precondiciones	• Click en opción de interfaz • Programa abierto y conectado a la base de datos.
Flujo principal	1. Sistema: carga datos actuales del salón 2. Usuario: ingresa datos nuevos del salón 3. Sistema: escribe datos
Flujo alternativo	
Poscondiciones	El salón es previsualizado.

Asignar salón

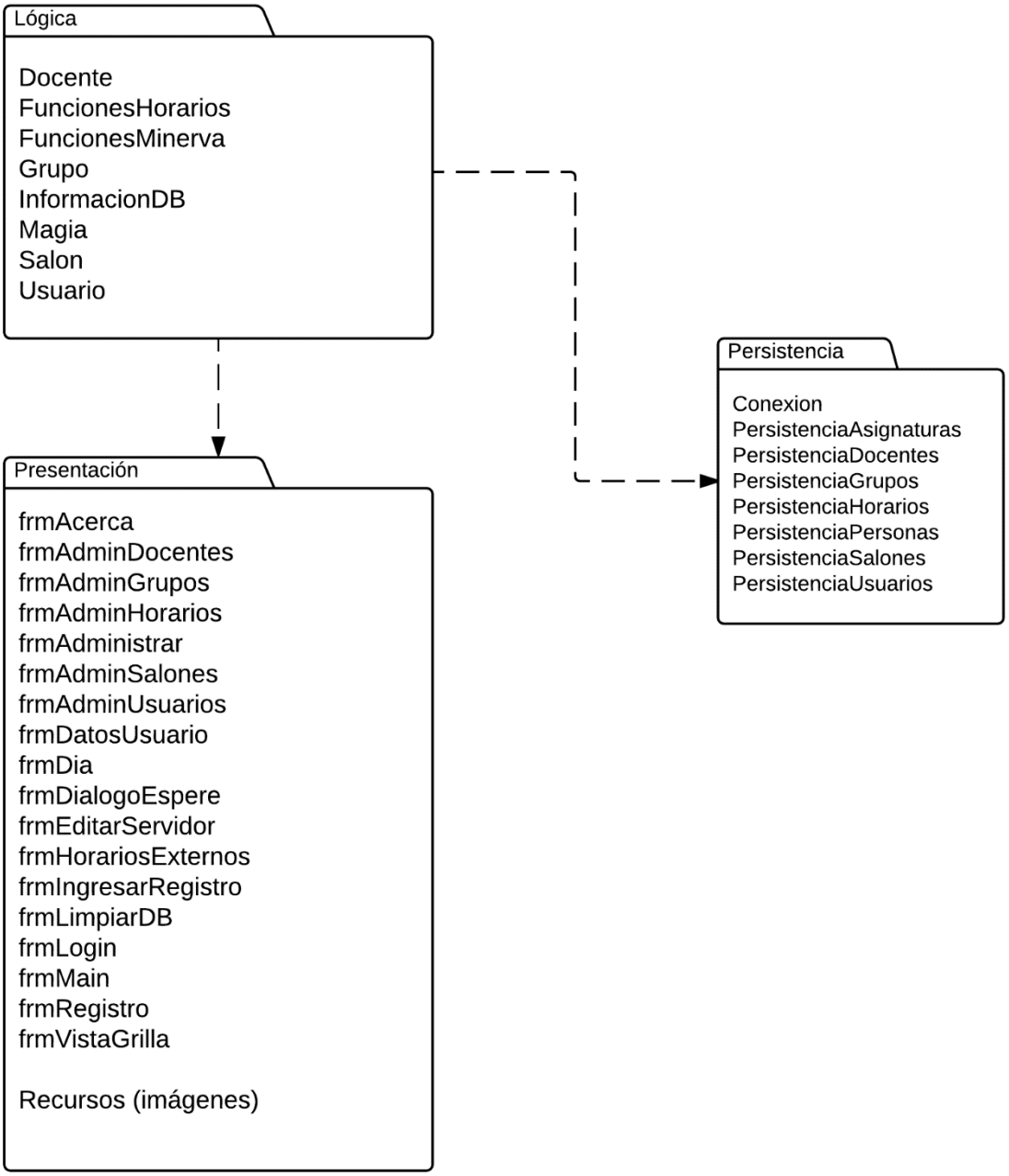
Nombre	Asignar salón a grupo
Actores	• Administrador • Funcionario • Base de datos MySQL
Descripción	Asigna un salón a un grupo. Se hace desde la vista de grupos.
Precondiciones	• Click en opción de interfaz • Programa abierto y conectado a la base de datos.

Flujo principal	1. Usuario: selecciona el salón a asignar a dicho grupo. 2. Sistema: guardar dicha selección en la base de datos.
Flujo alternativo	
Poscondiciones	El grupo es previsualizado.

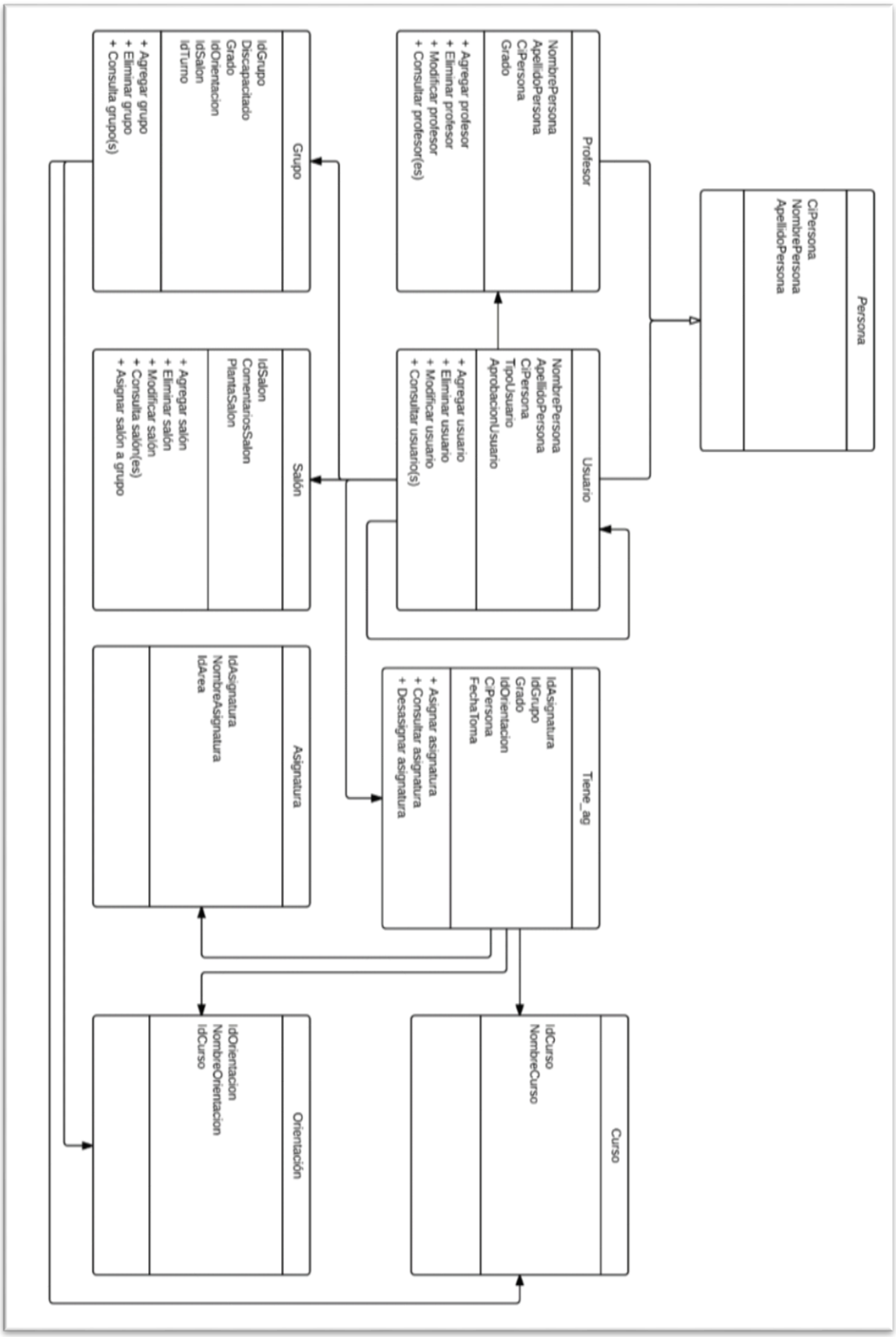
Previsualizar salón

Nombre	Previsualizar salón
Actores	• Administrador • Funcionario • Base de datos MySQL
Descripción	Carga los datos de un salón y los muestra en pantalla
Precondiciones	• Click en opción de interfaz • Programa abierto y conectado a la base de datos.
Flujo principal	1. Sistema: Carga y muestra los datos del salón.
Flujo alternativo	
Poscondiciones	Los datos son mostrados en pantalla.

4.7 Diagrama de componentes



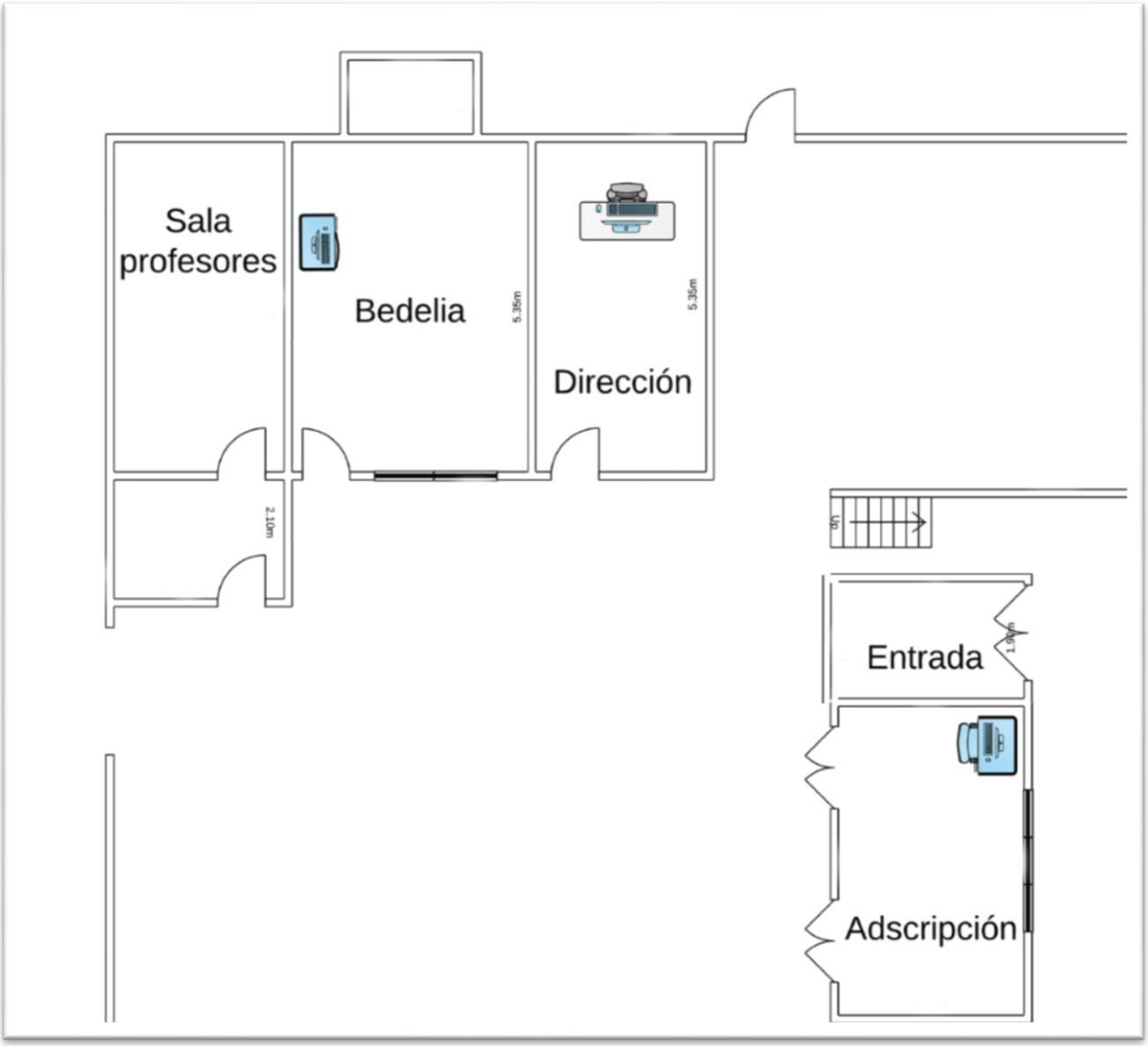
4.8 Diagrama de clases



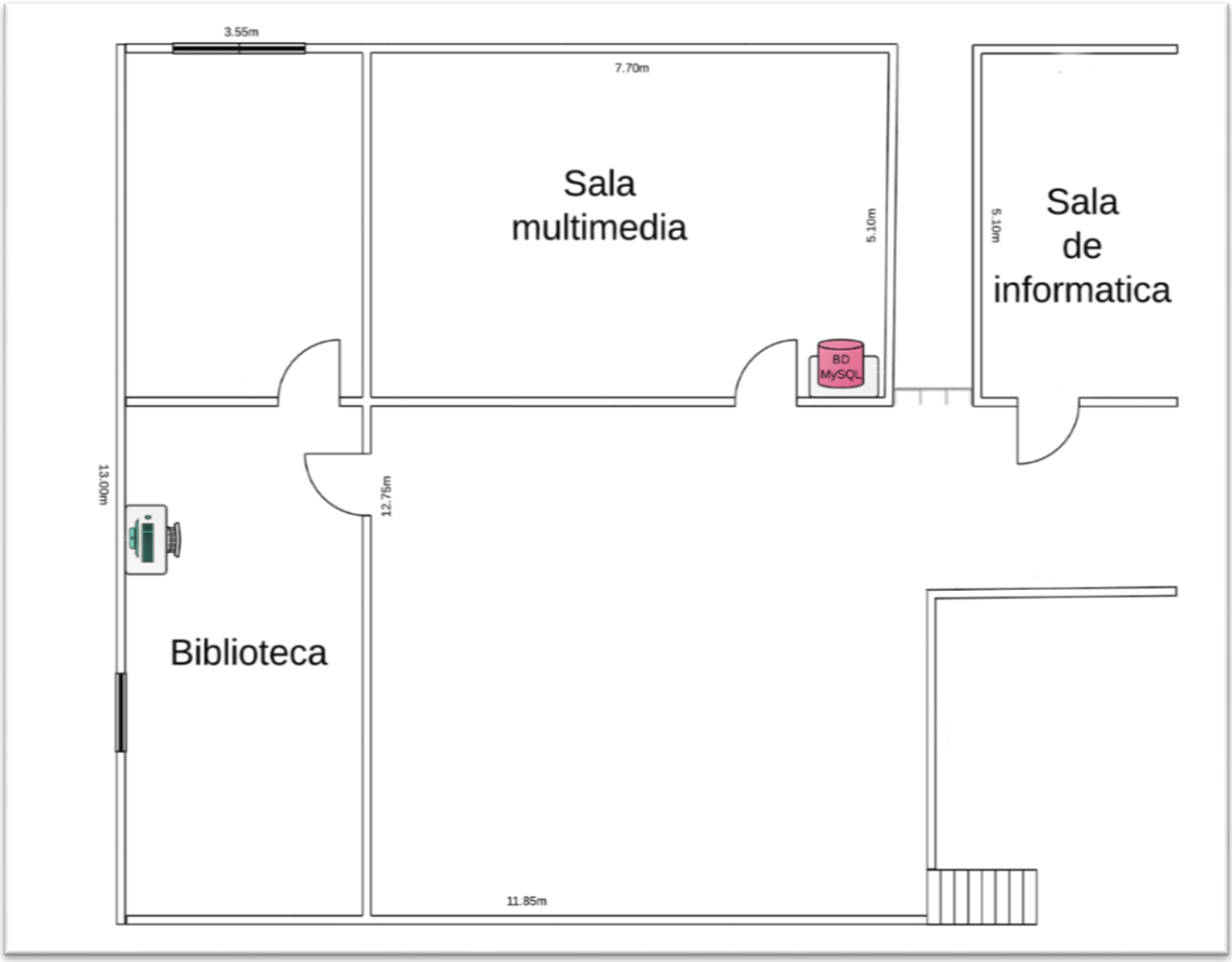
5. Planos y cableado

5.1 Planos de la institución

Primer piso



Segundo piso



5.2 Normas de cableado estructurado

MAITS Software se basa en una serie de normas a la hora de la realización del cableado, para así garantizar el óptimo funcionamiento de la red.

5.2.1 ¿Qué es un sistema de cableado estructurado?

El sistema de cableado estructurado (SCE) es una serie de estándares definidos por la TIA/EIA que determinan como:

- Diseñar
- Construir
- Administrar

Un sistema de cableado que es estructurado, el cual se refiere a todo el cableado y componentes instalados en una red basados en un orden lógico y organizado.

En la actualidad, los sistemas de Cableado Estructurado (CE) soportan una gran cantidad de servicios y aplicaciones (voz, datos, video, texto, imágenes) tales como:

- Teléfonos análogos y digitales
- Redes locales
- Sistemas de seguridad y vigilancia.

5.2.2 Importancia del cableado estructurado

- Mediante el seguimiento de las normas, se asegura que el cableado será flexible y seguro.
- La instalación es más sencilla e incluso más económica.

Organizaciones

Hay muchas organizaciones involucradas en el cableado estructurado en el mundo. En Estados Unidos es la ANSI, TIA e EIA, Internacionalmente es la ISO (International Standards Organization).

El propósito de las organizaciones de estándares es formular un conjunto de reglas comunes para todos en la industria. En el caso del cableado estructurado es proveer un conjunto estándar de reglas que permitan el soporte de múltiples marcas o fabricantes.

- Electronic Industries Alliance (EIA)

Organización conformada en 1997 por compañías electrónicas y de alta tecnología de los Estados Unidos.

- American National Standards Institute (ANSI)

Instituto Nacional Estadounidense de Estándares. Aparece el año 1928. Obtuvo su nombre actual en 1969.

- Telecommunications Industry Association (TIA)

Es una asociación de los Estados Unidos que representa casi 600 compañías de Telecomunicaciones

- . International Organization for Standarization (ISO)

Nace después de la segunda guerra mundial (1946). Su función principal es la de buscar la estandarización de normas de productos y seguridad para las empresas u organizaciones a nivel internacional

5.2.3 Normas

- ANSI/TIA/EIA 568-A

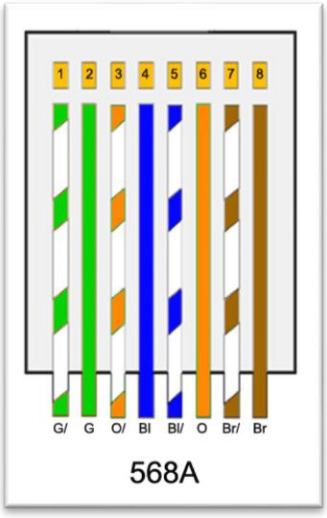
Hace Referencia al Cableado de Telecomunicaciones para Edificios Comerciales que Soporte un ambiente multiproducto y multiproveedor.

El propósito de esta norma es definir la planeación e instalación de cableado de edificios

Este estándar fue desarrollado y aprobado por comités del Instituto Nacional Americano de Normas (ANSI), la Asociación de la Industria de Telecomunicaciones (TIA), y la Asociación de la Industria Electrónica (EIA).

La norma establece criterios técnicos y de rendimiento para la instalación de cableado de edificios comerciales.

Según este estándar, la forma de engastar un cable UTP con un conector RJ-45 macho sigue el orden especificado en la tabla siguiente:

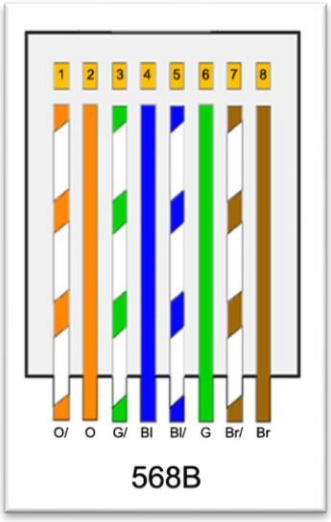


1. BLANCO-VERDE
2. VERDE
3. BLANCO-NARANJA
4. AZUL
5. BLANCO-AZUL
6. NARANJA
7. BLANCO-MARRÓN
8. MARRÓN

- ANSI/TIA/EIA 568-B

Estándar que define el cableado de telecomunicaciones para edificios comerciales. Desarrollado por AT&T con el nombre de 258A.

Posteriormente fue incluido y nombrado como TIA/EIA-568-B.Según este estándar, la forma de encajar el cable UTP con un conector RJ-45 macho sigue el orden especificado en la tabla siguiente:



1. BLANCO-NARANJA
2. NARANJA
3. BLANCO-VERDE
4. AZUL
5. BLANCO-AZUL
6. VERDE
7. BLANCO-MARRÓN
8. MARRÓN

6. Costo del sistema

6.1 Presupuestos

Cantidad	Descripción	Importador	Precio unitario	Precio total
4	Licencia Windows 7	NNET	U$S 185	U$S 740
100mts	Cable UTP cat5e	BOREAL	U$S 52	U$S 52
1	Switch	La Compra Perfecta	U$S 26	U$S 26
4	Computadoras	THOT Computación	U$S 260	U$S 1040
4	Monitores	La Oferta Irresistible	U$S 89,99	U$S 359,96
1	Raspberry Pi 2	Laaca Electrónica	U$S 65	U$S 65
100mts	Ductos	UruElectric	U$S 1.46	U$S 73
25	Fichas RJ45	Districomp	U$S 0,50	U$S 12,5
2	Cajas de Registro	UruElectric	U$S 23.14	U$S 46,28
				TOTAL: U$S 2414,74

**Todos los precios incluyen IVA.*

6.2 Especificaciones técnicas

Computadoras

- Intel Dual Core g3260 3,3ghz
- Mother GA-H81M-S1 Gigabyte con USB 3.0
- Memoria 4Gb DDR3
- Disco sata III 500gb
- Incluye kit de de periféricos (teclado, mouse y parlantes)

Servidor (Raspberry Pi 2)

- Procesador: 900MHz quad-core ARM Cortex-A7 CPU
- Memoria: 1GB
- 4 puertos USB
- 40 pines GPIO
- Puerto HDMI
- Puerto Ethernet

- microSD slot

Switch

- TPLINK TL-SG1005D
- 5 puertos RJ45 con velocidad de 10/100 Mbps
- Fuente de alimentación externa
- Adaptador de corriente externo 5.0VDC / 0.6A

Licencias

- Windows 7 PRO 64 Bits

Ductos

- Medidas: 20mm x 10mm
- Ductos autoadhesivos
- Largo: 2mts

Cable

- Bobina de 100mts de cable UTP categoría 5e

Registro

- IP 55 Estanco
- Línea Steck Logicbox
- Medidas: 30cm x 22cm x 12cm

Conectores

- RJ45 categoría 5e
- Conector de 8 vías

Monitores

- DELL 22''
- Refabricado grado A+
- Resolución 1680x1050

7. Métricas

COMPUTACIÒN DE MÈTRICAS DE PUNTO DE FUNCIONES.

	Factor de ponderación.							
Parámetro de medición	**Simples**		**Medio**		**Complejo**			**Total**
Número de entradas de usuario	0	x 3	34	x 4	45	x 6	=	406
Número de salidas de usuario	0	x 4	37	x 5	0	x 7	=	185
Número de peticiones de usuario	0	x 3	37	x 4	0	x 6	=	148
Número de archivos	0	x 7	0	x 10	15	x 15	=	225
Número de interfaces externas	0	x 5	0	x 7	2	x 10	=	20
Cuenta = Total								984

Nº de entradas de usuario	Los datos ingresados por el usuario.			
Nº de salidas de usuario	Informes, pantallas, mensajes de error .			
Nº de peticiones de usuario	Entradas interactivas			
Nº de archivos	Archivos maestro (lógico)			
Nº de interfaces externas	Todos los dispositivos que se utilicen para intercambiar datos.			

PF.= Cuenta-Total * (0,65+0,001* sumatoria de Fi)

Fi (i=1 a 14) son los valores de ajuste de complejidad.

AJUSTE DE COMPLEJIDAD

	0	1	2	3	4	5
	No influencia	**Incidental**	**Moderado**	**Medio**	**Significativo**	**Esencial**
1. ¿Requiere el sistema copias de seguridad y recup. fiables?		1				
2. ¿Se requiere comunicac. de datos ?						5
3.¿ Existen funciones de func. distribuido?	0					
4. ¿ Es crítico el rendimiento?			2			
5. ¿ Se ejecutará el sistema en un entorno operativo existente y fuertemente utilizado ?					4	
6- ¿ Requiere el sistema entrada de datos interactiva ?						5
7. ¿ Requiere la entrada de datos interactivas que las transac. de entrada se lleven a cabo sobre múltiples pantallas u operaciones ?				3		
8. ¿ Se actualizan los archivos maestro en forma interactiva ?						5
9. ¿ Son complejas las entradas, las salidas, los archivos o las peticiones?				3		
10. ¿ Es complejo el procesamiento interno ?						5
11. ¿ Se diseñará el código para ser reutilizable ?					4	
12. ¿ Están incluidas en el diseño la conversión y la instalación ?				3		
13. ¿ Se diseñará el sistema para múltiples instalaciones en diferentes organizaciones ?			2			0
14. ¿ Se diseñará la aplicación para facilitar los cambios y para ser fácilmente utilizada por el usuario ?						5
	0	1	4	9	8	25

Fi = 47

PF.= Cuenta-Total * (0,65+0,001* sumatoria de Fi) = 686

TAMAÑO DEL PROYECTO y COSTOS DEL PROYECTO.

LENGUAJE DE PROGRAMACIÓN		LDC/PF	TOTALES
Ensamblador		320	219471,36
C		128	87788,544
COBOL		105	72014,04
FORTRAN		105	72014,04
PASCAL		90	61726,32
ADA		70	48009,36
LENGUAJES ORIENTADO A OBJET.		30	20575,44
LENGUAJES DE 4a.GENERACION		20	13716,96
GENERADORES DE CÓDIGO		**15**	**10287,72**
HOJAS DE CÁLCULO		6	4115,088
LENGUAJES GRÁFICOS (ICONOS)		4	2743,392
VALOR ESPERADO = (OPTIM +4PROBABLE+PESIMISTA)/6			
VALOR ESPERADO =	**6070,453333**	lineas	

Según valores tomados de la bibliografía específica , se escriben 620 Líneas/mes por persona

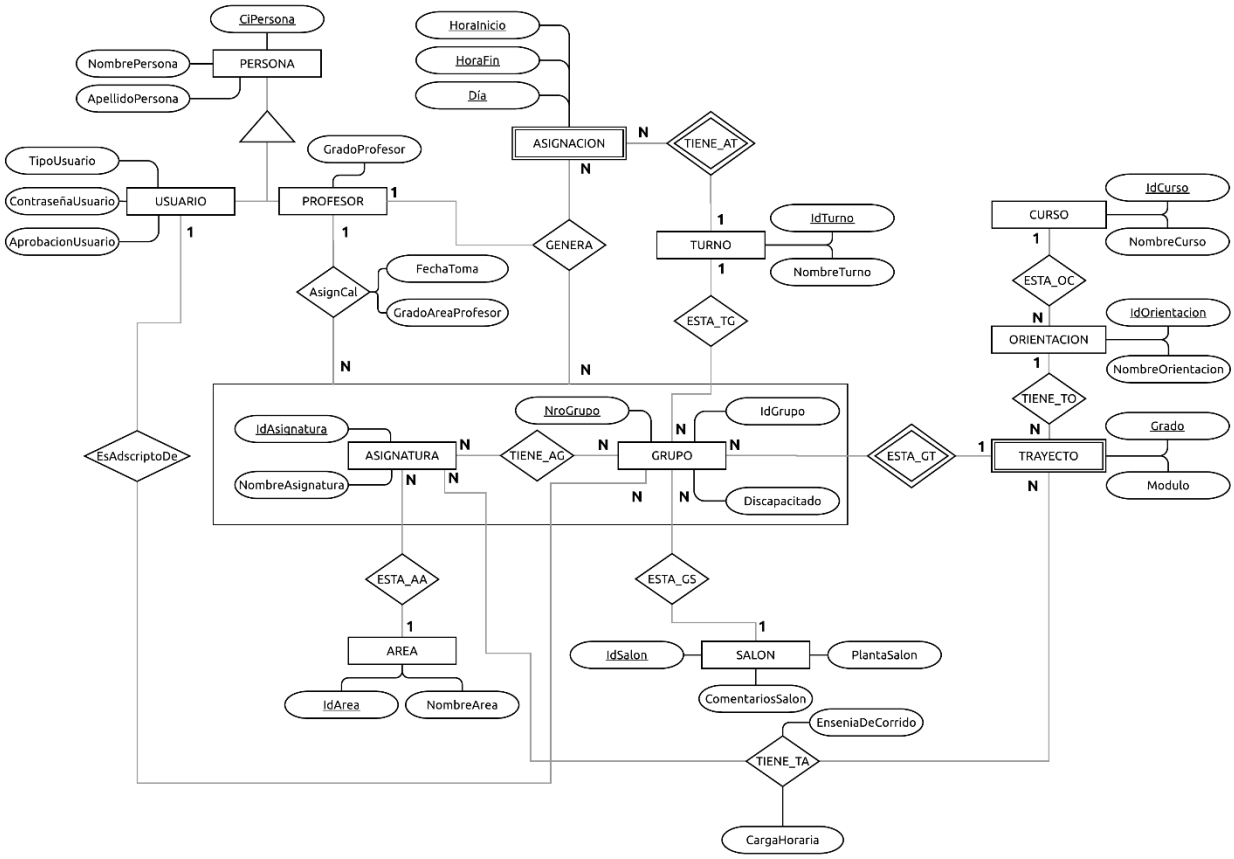
tiempo total = **9,79**

Costos= 29373,1613 dólares

Se consideró un salario de U$S 15/hora , y 200 horas por mes

8. Base de datos

8.1 Modelo Entidad Relación (MER)



8.2 Pasaje a tablas

PERSONA (**CiPersona**, NombrePersona, ApellidoPersona)

USUARIO (***CiPersona***, TipoUsuario, ContraseñaUsuario, AprobacionUsuario)

PROFESOR (***CiPersona***, GradoProfesor)

ASIGNATURA (**IdAsignatura**, NombreAsignatura, ***IdArea***)

GRUPO (**NroGrupo**, IdGrupo, Discapacitado, ***IdSalon***, ***IdTurno***, ***Grado***, ***IdOrientacion***, ***CiPersona***)

ASIGNACION (**HoraInicio**, **HoraFin**, **Dia**, ***IdTurno***)

SALON (**IdSalon**, PlantaSalon, ComentariosSalon)

TRAYECTO (**Grado**, Modulo, ***IdOrientacion***)

ORIENTACION (**IdOrientacion**, NombreOrientacion, ***IdCurso***)

CURSO (**IdCurso**, NombreCurso)

AREA (**IdArea**, NombreArea)

TURNO (**IdTurno**, NombreTurno)

GENERA (***IdAsignatura***, ***NroGrupo***, ***HoraInicio***, ***HoraFin***, ***Dia***, ***IdTurno***, ***CiPersona***)

TIENE_TA (***IdAsignatura***, ***Grado***, ***IdOrientacion***, CargaHoraria, EnseniaDeCorrido)

TIENE_AG (***IdAsignatura***, ***NroGrupo***, FechaToma, GradoAreaProfesor, ***CiPersona***)

8.3 Normalización

8.3.1 Primera Forma Normal (1FN)

Tabla	Atributos multivalorados y/o estructurados	Verifica 1FN
PERSONA	No presenta	✓
USUARIO	No presenta	✓
PROFESOR	No presenta	✓
ASIGNATURA	No presenta	✓
GRUPO	No presenta	✓
ASIGNACION	No presenta	✓
SALON	No presenta	✓
TRAYECTO	No presenta	✓
ORIENTACION	No presenta	✓
CURSO	No presenta	✓
AREA	No presenta	✓
TURNO	No presenta	✓
GENERA	No presenta	✓

TIENE_AG	No presenta	✓
TIENE_TA	No presenta	✓

- Las tablas **PERSONA, USUARIO, PROFESOR, ASIGNATURA, GRUPO, ASIGNACION, SALON, TRAYECTO, ORIENTACION, CURSO, AREA, TURNO, GENERA, TIENE_AG, TIENE_TA** están en 1FN porque todos sus atributos son atómicos e indivisibles.

8.3.2 Segunda Forma Normal (2FN)

Tabla	Dependencias	Verifica 2FN
PERSONA	**CiPersona** → NombrePersona, ApellidoPersona	✓
USUARIO	**CiPersona** → TipoUsuario, ContraseniaUsuario, AprobacionUsuario	✓
PROFESOR	**CiPersona** → GradoProfesor	✓
ASIGNATURA	**IdAsignatura** → NombreAsignatura, IdArea	✓
GRUPO	**NroGrupo** → IdGrupo, Discapacitado, IdSalon, IdTurno, Grado, IdOrientacion, CiPersona	✓
ASIGNACION	**HoraInicio, HoraFin, Dia, IdTurno** No hay dependencias.	✓
SALON	**IdSalon** → ComentariosSalon, PlantaSalon	✓
TRAYECTO	**Grado, IdOrientacion** → Modulo	✓
ORIENTACION	**IdOrientacion** → NombreOrientacion, IdCurso	✓
CURSO	**IdCurso** → NombreCurso	✓
AREA	**IdArea** → NombreArea	✓
TURNO	**IdTurno** → NombreTurno	✓
GENERA	**IdAsignatura, NroGrupo, HoraInicio, HoraFin, Dia, IdTurno** → CiPersona	✓
TIENE_AG	**IdAsignatura, NroGrupo** → FechaToma, GradoAreaProfesor, CiPersona	✓
TIENE_TA	**Grado, IdOrientacion, IdAsignatura** → CargaHoraria, EnseniaDeCorrido	✓

- Las tablas **PERSONA, USUARIO, PROFESOR, ASIGNATURA, GRUPO, ASIGNACION, SALON, TRAYECTO, ORIENTACION, CURSO, AREA, TURNO, GENERA, TIENE_AG, TIENE_TA** están en 2FN porque están en 1FN y todos sus atributos no primos dependen de forma total de la clave primaria.

8.3.3 Tercera Forma Normal (3FN)

Tabla	Dependencias	Verifica 3FN
PERSONA	Atributos no primos: NombrePersona, ApellidoPersona No hay dependencias.	✓
USUARIO	Atributos no primos: TipoUsuario, ContraseñaUsuario, AprobacionUsuario No hay dependencias.	✓
PROFESOR	Atributos no primos: GradoProfesor No hay dependencias.	✓
ASIGNATURA	Atributos no primos: NombreAsignatura, IdArea No hay dependencias.	✓
GRUPO	Atributos no primos: Discapacitado, IdSalon, IdTurno, IdOrientacion, CiPersona, Grado No hay dependencias.	✓
ASIGNACION	Atributos no primos: No posee.	✓
SALON	Atributos no primos: ComentariosSalon, PlantaSalon No hay dependencias.	✓
TRAYECTO	Atributos no primos: Modulo No hay dependencias.	✓
ORIENTACION	Atributos no primos: NombreOrientacion, IdCurso No hay dependencias.	✓
CURSO	Atributos no primos: NombreCurso No hay dependencias.	✓
AREA	Atributos no primos: NombreArea No hay dependencias.	✓
TURNO	Atributos no primos: NombreTurno No hay dependencias.	✓
GENERA	Atributos no primos: CiPersona No hay dependencias.	✓
TIENE_AG	Atributos no primos: FechaToma, GradoAreaProfesor, CiPersona No hay dependencias.	✓

TIENE_TA	Atributos no primos: CargaHoraria, EnseniaDeCorrido No hay dependencias.	✓

- Las tablas **PERSONA, USUARIO, PROFESOR, ASIGNATURA, GRUPO, ASIGNACION, SALON, TRAYECTO, ORIENTACION, CURSO, AREA, TURNO, GENERA, TIENE_AG, TIENE_TA** están en 3FN porque están en 2FN, por lo tanto también en 1FN, y además ningún atributo no primo depende transitivamente de la clave primaria.

8.4 Diccionario de datos

Tabla: PERSONA

Descripción: Almacena los datos referidos a las personas
Viene cargado con '-1', como dato defecto.

Columna	Tipo (largo)	Descripción
CiPersona	VARCHAR (8)	CI de la persona. Clave primaria.
NombrePersona	VARCHAR (25)	Nombre de la persona.
ApellidoPersona	VARCHAR (25)	Apellido de la persona.

Tabla: USUARIO

Descripción: Almacena los datos referidos a los usuarios
Viene cargado con '-1', utilizando éste valor como adscripto por defecto, es decir, Adscripto sin asignar.

Columna	Tipo (largo)	Descripción
CiPersona	VARCHAR (8)	CI de la persona. Clave primaria heredada de la tabla PERSONA.
TipoUsuario	VARCHAR (13)	Tipo de usuario.
ContraseniaUsuario	VARCHAR (25)	Contraseña del usuario.
AprobacionUsuario	BOOLEAN	¿Acceso aprobado?.

Tabla: PROFESOR

Descripción: Almacena los datos referidos a los profesores.
Viene cargado con '-1', utilizando éste valor como profesor por defecto, es decir, Profesor sin asignar.

Columna	Tipo (largo)	Descripción
CiPersona	INT (8)	CI del profesor. Clave primaria heredada de la tabla PERSONA.
GradoProfesor	INT (3)	Grado general del profesor.

Tabla: ASIGNATURA

Descripción: Almacena los datos referidos a las asignaturas. **
Viene cargado con '-1', utilizando éste valor como Asignatura sin asignar/definir.

Columna	Tipo (largo)	Descripción
IdAsignatura	VARCHAR (5)	Id de la asignatura. Clave primaria.
NombreAsignatura	VARCHAR (50)	Nombre de la asignatura.
IdArea	VARCHAR (4)	Id del área. Clave foránea de la tabla AREA.

Tabla: GRUPO

Descripción: Almacena los datos referidos a los grupos.

Columna	Tipo (largo)	Descripción
NroGrupo	INT	Número único (usado sólo a nivel de base de datos) de grupo. Clave primaria. Auto incrementa.
IdGrupo	VARCHAR (4)	ID del grupo.
Discapacitado	BOOLEAN	¿El grupo tiene un alumno discapacitado?
IdSalon	INT (2)	Id del salón. Clave foránea de la tabla SALON.
IdTurno	INT (2)	Id del turno. Clave foránea de la tabla TURNO.
Grado	INT (2)	Grado del grupo. Clave foránea de la tabla TRAYECTO.
IdOrientacion	VARCHAR (4)	Orientación del Grupo. Clave foránea de la tabla TRAYECTO.

CiPersona	VARCHAR (8)	Ci del Usuario Adscripto. Clave foránea de la tabla PERSONA.

Tabla: ASIGNACION

Descripción: Almacena las horas de entrada (a clase) y salida (de clase) de tal día en determinado turno. **

Columna	Tipo (largo)	Descripción
HoraInicio	TIME	Hora de inicio (de la clase). Clave primaria.
HoraFin	TIME	Hora de finalización (de la clase). Clave primaria.
Dia	VARCHAR (10)	Día de la semana. Clave primaria.
IdTurno	INT (2)	Id del turno. Clave foránea de la tabla TURNO.

Tabla: SALON

Descripción: Almacena los datos referidos a los salones.
Viene cargado con '-1', utilizando éste valor como salón no asignado.

Columna	Tipo (largo)	Descripción
IdSalon	INT (2)	Id del salón. Clave primaria.
ComentariosSalon	TEXT	Comentarios del salón.
PlantaSalon	VARCHAR (15)	Planta del salón.

Tabla: TRAYECTO

Descripción: Almacena los datos referidos a los trayectos. **

Columna	Tipo (largo)	Descripción
Grado	INT (2)	Grado del trayecto. Clave primaria.
Modulo	INT (2)	Modulo del trayecto.
IdOrientacion	VARCHAR (4)	Id de la orientación. Clave primaria foránea de la tabla ORIENTACION.

Tabla: ORIENTACION

Descripción: Almacena los datos referidos a las orientaciones. **

Columna	Tipo (largo)	Descripción
IdOrientacion	VARCHAR (4)	Id de la orientación. Clave primaria.
NombreOrientacion	VARCHAR (30)	Nombre de la orientación.
IdCurso	VARCHAR (4)	Id del curso. Clave foránea de la tabla CURSO.

Tabla: CURSO

Descripción: Almacena los datos referidos a los cursos. **

Columna	Tipo (largo)	Descripción
IdCurso	VARCHAR (4)	Id del curso. Clave primaria.
NombreCurso	VARCHAR (30)	Nombre del curso

Tabla: AREA

Descripción: Almacena los datos referidos a las áreas. **
Viene cargado con '-1', utilizando éste valor como Área sin definir.

Columna	Tipo (largo)	Descripción
IdArea	VARCHAR (4)	Id del área. Clave primaria.
NombreArea	VARCHAR (30)	Nombre del área.

Tabla: TURNO

Descripción: Almacena los datos referidos a los turnos. **

Columna	Tipo (largo)	Descripción
IdTurno	INT (2)	Id del turno. Clave primaria.
NombreTurno	VARCHAR (30)	Nombre del turno.

Tabla: GENERA

Descripción: Tomando en cuenta las claves primarias de ASIGNACION y las de la agregación de GRUPO + ASIGNATURA, se genera un calendario, cada fila de ésta tabla es una hora de clase.

Columna	Tipo (largo)	Descripción
IdAsignatura	VARCHAR (5)	ID de Asignatura. Clave primaria foránea de la tabla ASIGNATURA.
NroGrupo	INT	Número único de grupo. Clave primaria foránea de la tabla GRUPO.
HoraInicio	TIME	Hora de inicio (de la clase). Clave primaria foránea de la tabla ASIGNACION.
HoraFin	TIME	Hora de finalización (de la clase). Clave primaria foránea de la tabla ASIGNACION.
Dia	VARCHAR (10)	Dia. Clave primaria foránea de la tabla ASIGNACION.
IdTurno	INT (2)	Id del Turno. Clave primaria foránea de la tabla TURNO.
CiPersona	VARCHAR (8)	Ci de la persona. Clave foránea de la tabla PERSONA.

Tabla: TIENE_AG

Descripción: Almacena las materias que han sido tomadas por docentes.

Columna	Tipo (largo)	Descripción
IdAsignatura	VARCHAR (5)	Id de la asignatura. Clave primaria foránea de la tabla ASIGNATURA.
NroGrupo	INT	Número único de grupo. Clave primaria foránea de la tabla GRUPO.
FechaToma	DATE	Fecha en la que asignatura fue asignada a él docente.
GradoAreaProfesor	INT (2)	Grado de área del profesor.
CiPersona	VARCHAR (8)	Ci de la persona. Clave foránea de la tabla PERSONA.

Tabla: TIENE_TA

Descripción: Almacena la cantidad de horas de una asignatura que tiene un grado de una orientación y si la misma se enseña de corrido (ignorar módulos).

(Ej. La asignatura 3894 (BD 2), pertenece a la orientación 480 (Informática) y es enseñada en 2do Grado, unas 3 horas semanales, todas de corrido (no se puede enseñar en 2 horas un día y otras 1). **

Columna	Tipo (largo)	Descripción
IdAsignatura	VARCHAR (5)	Id de la asignatura. Clave primaria foránea de la tabla ASIGNATURA.
Grado	INT (2)	Grado. Clave primaria foránea de la tabla TRAYECTO.
IdOrientacion	VARCHAR (4)	Id de la orientación. Clave primaria foránea de la tabla TRAYECTO.
CargaHoraria	INT (2)	Carga horaria semanal de determinada asignatura.
EnseniaDeCorrido	BOOLEAN	Determina si todas las horas de la materia deben enseñar en un día, o se pueden separar en módulos de 2 horas.

** Cabe destacar que estos datos vienen precargados en la base de datos, y sólo se realizan consultas.

8.5 Vistas

Nombre de vista	Descripción
Calendario	Provee las horas a la que cada asignatura comienza
DatosGrupos	Es utilizada para obtener los datos de determinado grupo
Adscriptos	Lista de adscriptos

8.6 Consultas

- Mostrar los usuarios del sistema con todos sus datos (nombre, apellido, etc). Si tienen definido en el modelo diferentes tipos de usuario, ordenar por tipo de usuario.

```
SELECT Usuario.*, Persona.NombrePersona, Persona.ApellidoPe
rsona
FROM Usuario, Persona
```

```
WHERE Usuario.CiPersona = Persona.CiPersona
ORDER BY Usuario.TipoUsuario;
```

- Mostrar para cada curso, orientación y trayecto (respetando ese orden), sus asignaturas.

```
SELECT Curso.IdCurso, Orientacion.IdOrientacion, Trayecto.G
rado, Tiene_Ta.IdAsignatura
FROM Curso, Orientacion, Trayecto, Tiene_Ta
WHERE Orientacion.IdCurso=Curso.IdCurso

AND Trayecto.IdOrientacion=Orientacion.IdOrientacion
AND Tiene_Ta.IdOrientacion=Trayecto.IdOrientacion
AND Tiene_Ta.Grado=Trayecto.Grado;
```

- Mostrar todos los grupos y si tienen asignado salón, sus datos. Ordenar resultado por turno y salón. *(En nuestro programa, la representación de no tener un salón asignado es -1)*

```
SELECT CONCAT(Grado, ' ', IdGrupo) as "Grupo",
Salon.*, IdTurno
FROM Grupo, Salon
WHERE Grupo.IdSalon=Salon.IdSalon
ORDER BY IdTurno, IdSalon;
```

- Dado un docente mostrar su "calendario semanal", ordenado por día de la semana y hora, las asignaturas que debe dictar y el grupo al cual corresponda.

```
SELECT Dia, HoraInicio, Grupo, Materia
FROM Calendario
WHERE CiPersona=12345678;
```

- Dado un grupo mostrar su "calendario semanal", es decir por día de la semana y hora, sus asignaturas con sus respectivos docentes.

```
SELECT Dia, HoraInicio, Grupo, Materia, CONCAT (Persona.Nom
brePersona, " ", Persona.ApellidoPersona) as Profesor
FROM Calendario, Persona
```

```
WHERE Grupo="3
BG" and Persona.CiPersona=Calendario.CiPersona
ORDER BY Dia, HoraInicio;
```

8.7 Políticas de respaldo

El administrador del sistema podrá respaldar la base de datos a un archivo sql para luiego ser cargado por él mismo a través de herramientas diseñadas para ello. Se recomienda hacer un respaldo antes de editar datos críticos.

9. Código fuente

9.1 Capa de lógica

9.1.1 Docente.vb

```
Imports MySql.Data.MySqlClient

Public Class Docente

    Public Shared Sub CargarAreas(frm As frmAdminDocentes)
        frm.cmbArea.Items.Clear()

        Dim conexion As New Conexion()
        Dim IdGrupo As String =
frm.cmbGrupo.Text.Substring(frm.cmbGrupo.Text.IndexOf(" "),
frm.cmbGrupo.Text.IndexOf(" - ")).Trim()
        Dim Grado As String = frm.cmbGrupo.Text.Substring(0,
frm.cmbGrupo.Text.IndexOf(" ")).Trim()

        Dim IdOrientacion As String =
PersistenciaGrupos.GetOrientacion(IdGrupo, Grado)

        Dim resultadosPersistencia As Object =
InformacionDB.GetAreasOrientacionByGrado(IdOrientacion, Grado)
        Dim reader As MySqlDataReader = resultadosPersistencia(0)
        While reader.Read()
            frm.cmbArea.Items.Add(reader("IdArea").ToString() & " - "
& reader("NombreArea") & "")
        End While

        reader.Close()
        resultadosPersistencia(1).Close()
    End Sub

    Public Shared Sub CargarGrupos(frm As frmAdminDocentes)
        Dim resultadosPersistencia As Object =
InformacionDB.GetGrupos()
        Dim reader As MySqlDataReader = resultadosPersistencia(0)

        While reader.Read()
            frm.cmbGrupo.Items.Add(reader("Grado") & " " &
reader("IdGrupo") & " - " & reader("NombreTurno") & "")
        End While

        reader.Close()
        resultadosPersistencia(1).Close()
    End Sub
```

```
    Public Shared Sub CargarDocentes(frm As frmAdminDocentes)
        frm.pnlDocentes.Controls.Clear()
        frm.totalDocentes = 0
        frm.lblCantidadDocentes.Text = "(" +
frm.totalDocentes.ToString() + ")"

        Dim resultadosPersistencia As Object =
InformacionDB.GetDocentes()
        Dim reader As MySqlDataReader = resultadosPersistencia(0)

        While reader.Read()
            If reader("CiPersona").ToString().Equals("-1") Then
                Continue While
            End If
            frm.AgregarWidgetDocente(reader("CiPersona"),
reader("CiPersona").ToString() & ControlChars.NewLine &
reader("NombrePersona") & " " & reader("ApellidoPersona"))
        End While
        reader.Close()

        resultadosPersistencia(1).Close()
    End Sub

    Public Shared Sub ActualizarDB(frm As frmAdminDocentes)
        Dim Ci As String = frm.txtCI.Text
        Dim Nombre As String = frm.txtNombre.Text
        Dim Apellido As String = frm.txtApellido.Text
        Dim Grado As Integer = frm.numGrado.Value

        If frm.btnAgregarDocente.Text.StartsWith("Agregar docente")
Then
            Try
                PersistenciaPersonas.Add(Ci, Nombre, Apellido)
                PersistenciaDocentes.Add(Ci, Grado)
                MessageBox.Show("Docente agregado correctamente",
"Docente agregado", MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
                CargarDocentes(frm)
                frm.InterfazEditarAsignaturasDocente(Ci)
            Catch ex As Exception
                If ex.ToString().Contains("Duplicate") Then
                    MessageBox.Show("Ya existe un docente (o usuario!)
con esa CI.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
                Else
                    MessageBox.Show(ex.ToString(), "Error",
MessageBoxButtons.OK, MessageBoxIcon.Warning)
                End If
            End Try
        Else
            PersistenciaPersonas.Edit(Ci, Nombre, Apellido)
            PersistenciaDocentes.Edit(Ci, Grado)
            MessageBox.Show("Información de docente actualizada
correctamente", "Docente actualizado", MessageBoxButtons.OK,
MessageBoxIcon.Asterisk)
```

```
            CargarDocentes(frm)
            frm.InterfazPrevisualizarDocente(frm.txtCI.Text)
        End If
    End Sub

    Public Shared Sub CargarInfo(Ci As String, frm As
frmAdminDocentes)
        Dim resultadosPersistencia As Object =
PersistenciaDocentes.GetInfo(Ci)
        Dim reader As MySqlDataReader = resultadosPersistencia(0)

        While reader.Read()
            frm.txtCI.Text = reader("CiPersona")
            frm.txtNombre.Text = reader("NombrePersona")
            frm.txtApellido.Text = reader("ApellidoPersona")
            frm.numGrado.Value = reader("GradoProfesor")
        End While
        reader.Close()

        resultadosPersistencia(1).Close()
    End Sub

    Public Shared Sub EliminarDocente(Ci As String, Nombre As String,
frm As frmAdminDocentes)
        Dim resultadosPersistenciaBackup As Object =
PersistenciaDocentes.GetInfo(Ci)
        Dim readerBackup As MySqlDataReader =
resultadosPersistenciaBackup(0)

        Try
            PersistenciaDocentes.Del(Ci)
            PersistenciaPersonas.Del(Ci)
            CargarDocentes(frm)
            frm.InterfazNuevoDocente()
            MessageBox.Show("Docente '" + Nombre + "' eliminado.",
"Docente eliminado.", MessageBoxButtons.OK,
MessageBoxIcon.Information)
        Catch ex As Exception
            MessageBox.Show("El docente no se puede eliminar, ya que
tiene asignaturas asignadas.", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
            Try
                While readerBackup.Read()

PersistenciaDocentes.Add(readerBackup("CiPersona"),
readerBackup("GradoProfesor"))
                End While
            Catch exx As Exception
                MessageBox.Show(exx.ToString(), "Error!",
MessageBoxButtons.OK)
            End Try
        End Try

        readerBackup.Close()
```

```
        resultadosPersistenciaBackup(1).Close()
    End Sub

    ' **** Asignaturas ****

    Public Shared Sub EliminarAsignatura(frm As frmAdminDocentes)
        Dim NroGrupo As String =
PersistenciaGrupos.GetNroGrupo(frm.lstAsignaturas.SelectedItems.Item(0
).SubItems(1).Text)
        Dim IdAsignatura As String =
frm.lstAsignaturas.SelectedItems.Item(0).SubItems(0).Text
        Dim CiPersona As String = frm.txtCI.Text

        Try
            PersistenciaHorarios.Del(IdAsignatura, NroGrupo,
CiPersona)
            PersistenciaAsignaturas.DesAsignar(NroGrupo, IdAsignatura,
CiPersona)
            CargarAsignaturas(CiPersona, frm)
            frm.btnEliminarAsignatura.Visible = False
            MessageBox.Show("Asignatura eliminada.", "Asignatura
eliminada.", MessageBoxButtons.OK, MessageBoxIcon.Information)
        Catch ex As Exception
            MessageBox.Show(ex.ToString(), "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try

    End Sub

    Public Shared Sub CargarAsignaturas(Ci As String, frm As
frmAdminDocentes)
        frm.lstAsignaturas.Items.Clear()
        Dim resultadosPersistencia As Object =
PersistenciaAsignaturas.GetAsignadasDocente(Ci)

        Dim reader As MySqlDataReader = resultadosPersistencia(0)
        While reader.Read()
            Dim item As New ListViewItem
            item =
frm.lstAsignaturas.Items.Add(reader("IdAsignatura").ToString())
            item.SubItems.Add(reader("Grado") & " " &
reader("IdGrupo"))
            item.SubItems.Add(reader("FechaToma").ToString())
        End While
        reader.Close()

        resultadosPersistencia(1).Close()
    End Sub

    Public Shared Sub ActualizarDB_Asignatura(frm As frmAdminDocentes)
        Dim NroGrupo As String =
PersistenciaGrupos.GetNroGrupo(frm.cmbGrupo.Text.Substring(0,
frm.cmbGrupo.Text.IndexOf(" - ")))
```

```
        Dim IdAsignatura As String =
frm.cmbAsignatura.Text.Substring(0, frm.cmbAsignatura.Text.IndexOf(" -
")).Trim()
        Dim FechaAhora As DateTime = Now
        Dim FechaToma As String = FechaAhora.ToString("yyyy-MM-dd")
        Dim GradoAreaProfesor As Integer = frm.numGradoArea.Value
        Dim Ci As String = frm.txtCI.Text

        If PersistenciaGrupos.GetAsignaturaTomada(IdAsignatura,
NroGrupo) Then
            MessageBox.Show("Esa asignatura ya ha sido asignada al
grupo.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            Return
        End If

        Try
            PersistenciaAsignaturas.Asignar(IdAsignatura, NroGrupo,
FechaToma, GradoAreaProfesor, Ci)

            Dim resultadosPersistenciaConflicto As Object =
PersistenciaDocentes.GetConflictosAsignacion(Ci, IdAsignatura,
NroGrupo)
            If resultadosPersistenciaConflicto(0) Then
                MessageBox.Show("El docente ya tiene un grupo asignado
(" & resultadosPersistenciaConflicto(1) & ") en el horario de ésta
asignatura." & vbCrLf & "Se recomienda acomodar los horarios en la
vista de Horarios.", "Conflictos detectados", MessageBoxButtons.OK,
MessageBoxIcon.Warning)
            End If

            PersistenciaHorarios.Edit(Ci, IdAsignatura, NroGrupo)
            CargarAsignaturas(Ci, frm)
            frm.lblNuevoDocente.Text = "Editar asignaturas del
docente"
            frm.LimpiarDatosAsignatura()
        Catch ex As Exception
            MessageBox.Show(ex.ToString(), "Error",
MessageBoxButtons.OK)
        End Try

    End Sub

    Public Shared Sub CargarAsignaturasGrupo(frm As frmAdminDocentes)
        frm.cmbAsignatura.Items.Clear()

        Dim IdGrupo As String =
frm.cmbGrupo.Text.Substring(frm.cmbGrupo.Text.IndexOf(" "),
frm.cmbGrupo.Text.IndexOf(" - ")).Trim()
        Dim Grado As String = frm.cmbGrupo.Text.Substring(0,
frm.cmbGrupo.Text.IndexOf(" ")).Trim()
        Dim IdArea As String = frm.cmbArea.Text.Substring(0,
frm.cmbArea.Text.IndexOf(" - "))
```

```
        Dim resultadosPersistencia As Object =
InformacionDB.GetAsignaturasForGrupo(IdGrupo, Grado, IdArea)
        Dim reader As MySqlDataReader = resultadosPersistencia(0)

        While reader.Read()
            If reader("IdAsignatura").ToString().Equals("-1") Then
                Continue While
            End If


frm.cmbAsignatura.Items.Add(reader("IdAsignatura").ToString() & " - "
& reader("NombreAsignatura"))
        End While

        reader.Close()
        resultadosPersistencia(1).Close()
    End Sub

    Public Shared Sub CrearGrillaHorariosDocente(frm As
frmAdminDocentes)
        Dim Dias As Object = {"Lunes", "Martes", "Miércoles",
"Jueves", "Viernes", "Sábado"}
        Dim frmHorariosExterno As New frmHorariosExternos(Nothing,
frm.ParentForm, frm.txtNombre.Text & " " & frm.txtApellido.Text)

        For Turno As Integer = 1 To 3

            Dim Horarios(0) As String

            If Turno = 3 Then
                ReDim Horarios(5)
            Else
                ReDim Horarios(6)
            End If

            Dim Row_Primera(8) As String
            Dim Row_Segunda(8) As String
            Dim Row_Tercera(8) As String
            Dim Row_Cuarta(8) As String
            Dim Row_Quinta(8) As String
            Dim Row_Sexta(8) As String
            Dim Row_Extra(8) As String

            Dim Rows As Object = {Row_Primera, Row_Segunda,
Row_Tercera, Row_Cuarta, Row_Quinta, Row_Sexta, Row_Extra}

            Dim resultadosPersistenciaHorariosTurno As Object =
PersistenciaHorarios.GetHorariosTurno(Turno)
            Dim readerHorariosTurno As MySqlDataReader =
resultadosPersistenciaHorariosTurno(0)
            Dim pos As Integer = 0
            While readerHorariosTurno.Read()
                Horarios(pos) =
readerHorariosTurno("HoraInicio").ToString()
```

```
                Rows(pos)(0) =
readerHorariosTurno("HoraInicio").ToString()
                Rows(pos)(0) = Rows(pos)(0).Substring(0,
Rows(pos)(0).Length - 3)
                pos += 1
            End While
            readerHorariosTurno.Close()
            resultadosPersistenciaHorariosTurno(1).Close()

            Dim resultadosPersistencia As Object =
PersistenciaHorarios.GetForProfesor(frm.txtCI.Text, Turno)
            Dim reader As MySqlDataReader = resultadosPersistencia(0)

            While reader.Read()
                Dim PosHora As Integer = Array.IndexOf(Horarios,
reader("HoraInicio").ToString())
                Dim PosDia As Integer = Array.IndexOf(Dias,
reader("Dia"))
                Rows(PosHora)(PosDia + 1) = reader("Materia") & vbCrLf
& reader("NombreProfesor")
            End While

            reader.Close()
            resultadosPersistencia(1).Close()

            Dim primeraVacia As Boolean = True
            Dim segundaVacia As Boolean = True
            Dim terceraVacia As Boolean = True
            Dim cuartaVacia As Boolean = True
            Dim quintaVacia As Boolean = True
            Dim sextaVacia As Boolean = True
            Dim extraVacia As Boolean = True

            pos = 0
            For Each Horario In Row_Primera
                If (Not IsNothing(Horario) And Not
String.IsNullOrEmpty(Horario)) And Not pos = 0 Then
                    primeraVacia = False
                End If
                pos += 1
            Next

            pos = 0
            For Each Horario In Row_Segunda
                If (Not IsNothing(Horario) And Not
String.IsNullOrEmpty(Horario)) And Not pos = 0 Then
                    segundaVacia = False
                End If
                pos += 1
            Next

            pos = 0
            For Each Horario In Row_Tercera
```

```
                If (Not IsNothing(Horario) And Not
String.IsNullOrEmpty(Horario)) And Not pos = 0 Then
                    terceraVacia = False
                End If
                pos += 1
            Next

            pos = 0
            For Each Horario In Row_Cuarta
                If (Not IsNothing(Horario) And Not
String.IsNullOrEmpty(Horario)) And Not pos = 0 Then
                    cuartaVacia = False
                End If
                pos += 1
            Next

            pos = 0
            For Each Horario In Row_Quinta
                If (Not IsNothing(Horario) And Not
String.IsNullOrEmpty(Horario)) And Not pos = 0 Then
                    quintaVacia = False
                End If
                pos += 1
            Next

            pos = 0
            For Each Horario In Row_Sexta
                If (Not IsNothing(Horario) And Not
String.IsNullOrEmpty(Horario)) And Not pos = 0 Then
                    sextaVacia = False
                End If
                pos += 1
            Next

            pos = 0
            For Each Horario In Row_Extra
                If (Not IsNothing(Horario) And Not
String.IsNullOrEmpty(Horario)) And Not pos = 0 Then
                    extraVacia = False
                End If
                pos += 1
            Next

            If Not primeraVacia Then

frmHorariosExterno.Grilla.dgvMaterias.Rows.Add(Row_Primera)
            End If
            If Not segundaVacia Then

frmHorariosExterno.Grilla.dgvMaterias.Rows.Add(Row_Segunda)
            End If
            If Not terceraVacia Then

frmHorariosExterno.Grilla.dgvMaterias.Rows.Add(Row_Tercera)
```

```
            End If
            If Not cuartaVacia Then

frmHorariosExterno.Grilla.dgvMaterias.Rows.Add(Row_Cuarta)
            End If
            If Not quintaVacia Then

frmHorariosExterno.Grilla.dgvMaterias.Rows.Add(Row_Quinta)
            End If
            If Not sextaVacia Then

frmHorariosExterno.Grilla.dgvMaterias.Rows.Add(Row_Sexta)
            End If
            If Not extraVacia Then

frmHorariosExterno.Grilla.dgvMaterias.Rows.Add(Row_Extra)
            End If
        Next
        frmHorariosExterno.ShowDialog(frm)
    End Sub
End Class
```

9.1.2 FuncionesHorarios.vb

```
Imports MySql.Data.MySqlClient

Public Class FuncionesHorarios
    Public Shared Sub CargarGrupos(frm As frmAdminHorarios)
        frm.cmbGrupo.Items.Clear()
        frm.cmbGrupo.Items.Add("...")
        frm.cmbGrupo.SelectedIndex = 0
        Dim resultadosPersistencia As Object =
InformacionDB.GetGrupos()

        Dim reader As MySqlDataReader = resultadosPersistencia(0)
        While reader.Read()
            frm.cmbGrupo.Items.Add(reader("Grado").ToString() + " " +
reader("IdGrupo").ToString())
        End While
        reader.Close()

        resultadosPersistencia(1).Close()
    End Sub

    Public Shared Sub CargarAsignaturas(frm As frmAdminHorarios)
        frm.Cursor = Cursors.WaitCursor
        frm.pnlMaterias.Enabled = False

        frm.LimpiarTablas()

        Dim NroGrupo As String =
PersistenciaGrupos.GetNroGrupo(frm.cmbGrupo.Text)

        Dim Tablas As Object = {frm.tableLunes, frm.tableMartes,
frm.tableMiercoles, frm.tableJueves, frm.tableViernes,
frm.tableSabado}
        For Each Tabla In Tablas
            Tabla.Enabled = False
        Next

        Dim BotonesAsignaturas As New List(Of List(Of Object))
        Dim Asignaturas As New List(Of String)

        Dim resultadosPersistenciaAsignaturas As Object =
PersistenciaAsignaturas.GetAllForGrupo(NroGrupo)

        Dim readerAsignaturas As MySqlDataReader =
resultadosPersistenciaAsignaturas(0)
        While readerAsignaturas.Read()
            For Carga As Integer = 1 To
Integer.Parse(readerAsignaturas("CargaHoraria"))
                Dim Asignatura As New Button
                AddHandler Asignatura.MouseEnter, AddressOf
frm.fixScroll
```

```
                AddHandler Asignatura.MouseWheel, AddressOf
frm.fixScroll
                Asignatura.TabStop = False
                Asignatura.Cursor = Cursors.Hand

                If readerAsignaturas("IdAsignatura").Equals("-1") Then
                    Continue For
                End If

                Dim DatosProfesor As Object =
PersistenciaAsignaturas.GetProfesor(readerAsignaturas("IdAsignatura"),
NroGrupo)
                Dim NombreProfesor As String = DatosProfesor(0)
                Dim CiProfesor As String = DatosProfesor(1)

                If NombreProfesor.Equals("-1") Then
                    NombreProfesor = "Sin profesor"
                End If

                Asignatura.Text =
readerAsignaturas("NombreAsignatura") & vbCrLf & NombreProfesor
                Asignatura.Tag = {readerAsignaturas("IdAsignatura"),
CiProfesor}

                Asignatura.Size = New Size(144, 60)
                Asignatura.BackColor = Color.White
                Asignatura.FlatStyle = FlatStyle.Flat
                Asignatura.FlatAppearance.BorderColor = Color.Red
                Asignatura.FlatAppearance.MouseDownBackColor =
Color.White
                Asignatura.FlatAppearance.MouseOverBackColor =
Color.White

                If
Asignaturas.Contains(readerAsignaturas("NombreAsignatura")) Then

BotonesAsignaturas(Asignaturas.IndexOf(readerAsignaturas("NombreAsigna
tura"))).Add(Asignatura)
                Else

Asignaturas.Add(readerAsignaturas("NombreAsignatura"))
                    BotonesAsignaturas.Add(New List(Of Object))

BotonesAsignaturas(Asignaturas.IndexOf(readerAsignaturas("NombreAsigna
tura"))).Add(Asignatura)
                End If

                AddHandler Asignatura.MouseDown, AddressOf
frm.Materia_MouseDown
                frm.pnlMaterias.Controls.Add(Asignatura)
            Next
        End While

        readerAsignaturas.Close()
```

```
        resultadosPersistenciaAsignaturas(1).Close()

        Dim Dias() As String = {"Lunes", "Martes", "Miércoles",
"Jueves", "Viernes", "Sábado"}
        Dim TablasLunes As Object = {frm.tableLunes1, frm.tableLunes2,
frm.tableLunes3, frm.tableLunes4, frm.tableLunes5, frm.tableLunes6,
frm.tableLunes7}
        Dim TablasMartes As Object = {frm.tableMartes1,
frm.tableMartes2, frm.tableMartes3, frm.tableMartes4,
frm.tableMartes5, frm.tableMartes6, frm.tableMartes7}
        Dim TablasMiercoles As Object = {frm.tableMiercoles1,
frm.tableMiercoles2, frm.tableMiercoles3, frm.tableMiercoles4,
frm.tableMiercoles5, frm.tableMiercoles6, frm.tableMiercoles7}
        Dim TablasJueves As Object = {frm.tableJueves1,
frm.tableJueves2, frm.tableJueves3, frm.tableJueves4,
frm.tableJueves5, frm.tableJueves6, frm.tableJueves7}
        Dim TablasViernes As Object = {frm.tableViernes1,
frm.tableViernes2, frm.tableViernes3, frm.tableViernes4,
frm.tableViernes5, frm.tableViernes6, frm.tableViernes7}
        Dim TablasSabado As Object = {frm.tableSabado1,
frm.tableSabado2, frm.tableSabado3, frm.tableSabado4,
frm.tableSabado5, frm.tableSabado6, frm.tableSabado7}
        Dim TablasDias As Object = {TablasLunes, TablasMartes,
TablasMiercoles, TablasJueves, TablasViernes, TablasSabado}
        Dim Horas As Object = {frm.horarioPrimera, frm.horarioSegunda,
frm.horarioTercera, frm.horarioCuarta, frm.horarioQuinta,
frm.horarioSexta, frm.horarioExtra}

        Dim resultadosPersistenciaCalendario As Object =
PersistenciaHorarios.GetCalendarioForGrupo(frm.cmbGrupo.Text)
        Dim readerCalendario As MySqlDataReader =
resultadosPersistenciaCalendario(0)
        While readerCalendario.Read()
            If readerCalendario("Materia").Equals("Sin asignar") Then
                Continue While
            End If

            Try
                Dim Index As Integer =
Asignaturas.IndexOf(readerCalendario("Materia"))
                Dim BotonAsignatura As Control =
BotonesAsignaturas(Index).ToArray().Last()

BotonesAsignaturas(Index).RemoveAt(BotonesAsignaturas(Index).Count -
1)
                Dim Dia As String = readerCalendario("Dia")

                Dim TablasDia As Object =
TablasDias(Array.IndexOf(Dias, Dia))
                Dim Hora As Integer = Array.IndexOf(Horas,
readerCalendario("HoraInicio").ToString())
                Dim Tabla As Object = TablasDia(Hora)
                If Not (Tabla.Controls.Count > 0) Then
                    frm.pnlMaterias.Controls.Remove(BotonAsignatura)
```

```
                    Tabla.Controls.Add(BotonAsignatura)
                End If
            Catch ex As Exception
            End Try
        End While

        readerCalendario.Close()
        resultadosPersistenciaCalendario(1).Close()

        frm.Cursor = Cursors.Default
        frm.pnlMaterias.Enabled = True

        For Each Tabla In Tablas
            Tabla.Enabled = True
        Next
    End Sub

    Public Shared Sub CargarHorariosTurno(frm As frmAdminHorarios)
        Dim IdTurno As String =
PersistenciaGrupos.GetTurno(frm.cmbGrupo.Text)

        Dim resultadosPersistencia As Object =
PersistenciaHorarios.GetHorariosTurno(IdTurno)
        Dim reader As MySqlDataReader = resultadosPersistencia(0)
        Dim posActual As Integer = 1
        While reader.Read()
            If posActual = 1 Then
                frm.horarioPrimera = reader("HoraInicio").ToString()
                frm.finPrimera = reader("HoraFin").ToString()
            ElseIf posActual = 2 Then
                frm.horarioSegunda = reader("HoraInicio").ToString()
                frm.finSegunda = reader("HoraFin").ToString()
            ElseIf posActual = 3 Then
                frm.horarioTercera = reader("HoraInicio").ToString()
                frm.finTercera = reader("HoraFin").ToString()
            ElseIf posActual = 4 Then
                frm.horarioCuarta = reader("HoraInicio").ToString()
                frm.finCuarta = reader("HoraFin").ToString()
            ElseIf posActual = 5 Then
                frm.horarioQuinta = reader("HoraInicio").ToString()
                frm.finQuinta = reader("HoraFin").ToString()
            ElseIf posActual = 6 Then
                frm.horarioSexta = reader("HoraInicio").ToString()
                frm.finSexta = reader("HoraFin").ToString()
            ElseIf posActual = 7 Then
                frm.horarioExtra = reader("HoraInicio").ToString()
                frm.finExtra = reader("HoraFin").ToString()
            End If
            posActual += 1
        End While

        If IdTurno.Equals("3") Then
            frm.pnlBordeOculto.Visible = True
            frm.pnlOcultarExtra.Visible = True
```

```
        Else
            frm.pnlBordeOculto.Visible = False
            frm.pnlOcultarExtra.Visible = False
        End If
        frm.actHorarios()

        reader.Close()
        resultadosPersistencia(1).Close()
    End Sub

    Public Shared Sub GuardarHorarios(frm As frmAdminHorarios)
        Dim NroGrupo As String =
PersistenciaGrupos.GetNroGrupo(frm.cmbGrupo.Text)
        Dim IdTurno As String =
PersistenciaGrupos.GetTurno(frm.cmbGrupo.Text)

        frm.frmAdministrar.habilitarBotones(False)

        Dim Tablas As Object = {frm.tableLunes, frm.tableMartes,
frm.tableMiercoles, frm.tableJueves, frm.tableViernes,
frm.tableSabado}
        For Each Tabla In Tablas
            Tabla.Enabled = False
        Next

        Dim Dias() As String = {"Lunes", "Martes", "Miércoles",
"Jueves", "Viernes", "Sábado"}
        Dim TablasLunes As Object = {frm.tableLunes1, frm.tableLunes2,
frm.tableLunes3, frm.tableLunes4, frm.tableLunes5, frm.tableLunes6,
frm.tableLunes7}
        Dim TablasMartes As Object = {frm.tableMartes1,
frm.tableMartes2, frm.tableMartes3, frm.tableMartes4,
frm.tableMartes5, frm.tableMartes6, frm.tableMartes7}
        Dim TablasMiercoles As Object = {frm.tableMiercoles1,
frm.tableMiercoles2, frm.tableMiercoles3, frm.tableMiercoles4,
frm.tableMiercoles5, frm.tableMiercoles6, frm.tableMiercoles7}
        Dim TablasJueves As Object = {frm.tableJueves1,
frm.tableJueves2, frm.tableJueves3, frm.tableJueves4,
frm.tableJueves5, frm.tableJueves6, frm.tableJueves7}
        Dim TablasViernes As Object = {frm.tableViernes1,
frm.tableViernes2, frm.tableViernes3, frm.tableViernes4,
frm.tableViernes5, frm.tableViernes6, frm.tableViernes7}
        Dim TablasSabado As Object = {frm.tableSabado1,
frm.tableSabado2, frm.tableSabado3, frm.tableSabado4,
frm.tableSabado5, frm.tableSabado6, frm.tableSabado7}
        Dim TablasDias As Object = {TablasLunes, TablasMartes,
TablasMiercoles, TablasJueves, TablasViernes, TablasSabado}
        Dim HorasInicio As Object = {frm.horarioPrimera,
frm.horarioSegunda, frm.horarioTercera, frm.horarioCuarta,
frm.horarioQuinta, frm.horarioSexta, frm.horarioExtra}
        Dim HorasFin As Object = {frm.finPrimera, frm.finSegunda,
frm.finTercera, frm.finCuarta, frm.finQuinta, frm.finSexta,
frm.finExtra}
```

```
        Dim HuboError As Boolean = False

        For Each Dia As String In Dias
            Dim TablasDia As Object = TablasDias(Array.IndexOf(Dias,
Dia))

            For Each HoraStr As String In HorasInicio
                Dim Hora As Integer = Array.IndexOf(HorasInicio,
HoraStr)
                Dim HoraInicio As String = HorasInicio(Hora)
                Dim HoraFin As String = HorasFin(Hora)

                Dim Tabla As TableLayoutPanel = TablasDia(Hora)

                Dim Btn As New Button
                AddHandler Btn.MouseEnter, AddressOf frm.fixScroll
                AddHandler Btn.MouseWheel, AddressOf frm.fixScroll

                Try
                    Btn = Tabla.Controls(0)
                Catch ex As Exception
                    Btn.Tag = {"-1", "-1"}
                End Try

                Dim resultadosConflicto As Object =
PersistenciaDocentes.GetConflictoGuardado(HoraInicio, Btn.Tag(1), Dia,
frm.cmbGrupo.Text, Btn.Tag(0))
                If resultadosConflicto(0) And Not
Btn.Tag(1).ToString().Equals("-1") Then
                    If Not HuboError Then
                        MessageBox.Show("El profesor '" &
resultadosConflicto(1) & "' ya enseña una materia al grupo " &
resultadosConflicto(2) & " el día: " & Dia & " a la hora " &
HoraInicio, "Error!", MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
                    End If
                    HuboError = True
                    Continue For
                End If

                PersistenciaHorarios.ForceDel(HoraInicio, HoraFin,
Dia, NroGrupo, IdTurno)
                PersistenciaHorarios.Add(Btn.Tag(0), NroGrupo,
HoraInicio, HoraFin, Dia, IdTurno, Btn.Tag(1))
            Next

            If HuboError Then
                MessageBox.Show("Se encontró al menos un error. Los
cambios que ha realizado, no han quedado guardados.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Stop)
                Exit For
            End If
        Next
        frm.Cursor = Cursors.Default
```

```
        For Each Tabla In Tablas
            Tabla.Enabled = True
        Next

        frm.dialogoEspere.SendToBack()
        frm.frmAdministrar.habilitarBotones(True)
        If Not HuboError Then
            FuncionesHorarios.CargarAsignaturas(frm)
            MessageBox.Show("Asignación de horarios guardada", "Todo
salió bien", MessageBoxButtons.OK, MessageBoxIcon.Information)
            frm.frmMain.RefrescarHorarios()
        End If
    End Sub
End Class
```

9.1.3 FuncionesMinerva.vb

```
Imports MySql.Data.MySqlClient
Public Class FuncionesMinerva

    Public Shared Sub CargarNombre(frm As frmMain)
        If frm.cuentaInvitado Then
            Return
        End If
        Dim InfoUsuario As Object =
PersistenciaUsuarios.GetNombre(frm.NombreUsuario)
        Dim Nombre As String = InfoUsuario(0)
        Dim Apellido As String = InfoUsuario(1)

        If String.IsNullOrEmpty(Nombre) Or
String.IsNullOrEmpty(Apellido) Then
            Try
                Dim x As New frmDatosUsuario(frm)
                x.ShowDialog(frm)
                CargarNombre(frm)
            Catch ex As Exception
                ' El usuario presionó salir.
            End Try
        Else
            frm.lblUsuario.Text = "Bienvenido " & Nombre & " " &
Apellido & "."
        End If
    End Sub

    Public Shared Sub CrearMenuFiltrado(frm As frmMain)
        frm.cmsFiltrado.Items.Clear()
        Dim Item, Turno, Curso As ToolStripMenuItem
        Turno = New ToolStripMenuItem()
        Turno.Text = "Turno"

        Curso = New ToolStripMenuItem()
        Curso.Text = "Curso"

        Dim resultadosPersistenciaTurnos As Object =
InformacionDB.GetTurnos()
        Dim readerTurno As MySqlDataReader =
resultadosPersistenciaTurnos(0)
        While readerTurno.Read()
            Item = New ToolStripMenuItem()
            Item.Text = readerTurno("NombreTurno") & " (" &
readerTurno("IdTurno") & ")"
            AddHandler Item.Click, AddressOf frm.filtroTurnoCambiado
            Turno.DropDownItems.Add(Item)
        End While

        readerTurno.Close()
        resultadosPersistenciaTurnos(1).Close()
```

```
        Dim resultadosPersistenciaCursos As Object =
InformacionDB.GetCursos()
        Dim readerCurso As MySqlDataReader =
resultadosPersistenciaCursos(0)
        While readerCurso.Read()
            Item = New ToolStripMenuItem()
            Item.Text = readerCurso("NombreCurso") & " (" &
readerCurso("IdCurso") & ")"
            AddHandler Item.Click, AddressOf frm.filtroCursoCambiado
            Curso.DropDownItems.Add(Item)
        End While

        readerCurso.Close()
        resultadosPersistenciaCursos(1).Close()
        frm.cmsFiltrado.Items.Add(Turno)
        frm.cmsFiltrado.Items.Add(Curso)
    End Sub

    Public Shared Sub CargarGrupos(frm As frmMain)
        Dim CursoElegido As String = Nothing
        Dim TurnoElegido As String = Nothing

        frm.cboGrupo.Items.Clear()
        frm.cboGrupo.Items.Add("Elija un grupo")
        frm.cboGrupo.SelectedIndex = 0

        If Not IsNothing(frm.cursoElegido) And Not
IsNothing(frm.turnoElegido) Then
            CursoElegido = frm.cursoElegido.Text.ToString()
            CursoElegido =
CursoElegido.Substring(CursoElegido.IndexOf(" (") + 2).Trim(")")

            TurnoElegido = frm.turnoElegido.Text.ToString()
            TurnoElegido =
TurnoElegido.Substring(TurnoElegido.IndexOf(" (") + 2).Trim(")")

        ElseIf Not IsNothing(frm.cursoElegido) Then
            CursoElegido = frm.cursoElegido.Text.ToString()
            CursoElegido =
CursoElegido.Substring(CursoElegido.IndexOf(" (") + 2).Trim(")")

        ElseIf Not IsNothing(frm.turnoElegido) Then
            TurnoElegido = frm.turnoElegido.Text.ToString()
            TurnoElegido =
TurnoElegido.Substring(TurnoElegido.IndexOf(" (") + 2).Trim(")")
        End If

        Dim resultadosPersistencia As Object =
InformacionDB.GetGrupos(CursoElegido, TurnoElegido)
        Dim reader As MySqlDataReader = resultadosPersistencia(0)

        While reader.Read()
            frm.cboGrupo.Items.Add(reader("Grado").ToString() & " " &
reader("IdGrupo"))
```

```
        End While

        reader.Close()
        resultadosPersistencia(1).Close()
    End Sub

    Public Shared Sub CargarInfoGrupo(frm As frmMain)
        frm.tblMaterias.Controls.Clear()
        frm.Cursor = Cursors.WaitCursor

        Dim IdOrientacion As String = ""
        Dim Grado As String = ""
        Dim IdGrupo As String = ""
        Dim NroGrupo As String =
PersistenciaGrupos.GetNroGrupo(frm.cboGrupo.Text)

        Dim resultadosPersistenciaDetalles As Object =
PersistenciaGrupos.GetDetalles(NroGrupo)
        Dim reader As MySqlDataReader =
resultadosPersistenciaDetalles(0)
        While reader.Read()
            frm.lblValorTipoCurso.Text = reader("Curso")
            frm.lblValorTipoSalon.Text = reader("Salon")
            If frm.lblValorTipoSalon.Text.Equals("-1") Then
                frm.lblValorTipoSalon.Text = "Sin asignar"
            End If
            frm.lblValorTipoGrado.Text = reader("Grado")
            frm.lblValorTipoTurno.Text = reader("NombreTurno")
            frm.lblValorTipoTurno.Tag = reader("IdTurno").ToString()
            frm.lblValorTipoAdscripto.Text = reader("Adscripto")
        End While

        reader.Close()
        resultadosPersistenciaDetalles(1).Close()

        Dim resultadosPersistenciaMaterias As Object =
PersistenciaAsignaturas.GetAllForGrupo(NroGrupo)
        Dim readerMaterias As MySqlDataReader =
resultadosPersistenciaMaterias(0)
        Dim pos As Integer = 1

        While readerMaterias.Read()
            Dim NombreAsignatura As String =
readerMaterias("NombreAsignatura")
            Dim NombreProfesor As String =
PersistenciaAsignaturas.GetProfesor(readerMaterias("IdAsignatura"),
NroGrupo)(0)

            Dim lblMateria As New Label
            Dim lblProfesor As New Label

            lblMateria.Font = New Font("Microsoft Sans Serif", 12,
FontStyle.Bold)
            lblProfesor.Font = New Font("Microsoft Sans Serif", 12)
```

```
            lblProfesor.Padding = New Padding(0, 0, 0, 5)
            lblMateria.ForeColor = Color.White

            lblMateria.Text = NombreAsignatura
            lblProfesor.Text = "         " & NombreProfesor


            frm.tblMaterias.RowStyles.Add(New
RowStyle(SizeType.AutoSize, 0))
            frm.tblMaterias.RowStyles.Add(New
RowStyle(SizeType.AutoSize, 0))
            frm.tblMaterias.Controls.Add(lblMateria, 0, pos)
            frm.tblMaterias.Controls.Add(lblProfesor, 0, pos + 1)

            lblMateria.AutoSize = True
            lblProfesor.AutoSize = True
            pos += 2
        End While

        readerMaterias.Close()

        resultadosPersistenciaMaterias(1).Close()
        frm.Cursor = Cursors.Default
    End Sub

    Public Shared Sub CargarHorariosGrupo(frm As frmMain)
        Dim Dias As Object = {"Lunes", "Martes", "Miércoles",
"Jueves", "Viernes", "Sábado"}
        frm.Cursor = Cursors.WaitCursor

        Dim Horarios(0) As String

        If frm.lblValorTipoTurno.Tag.Equals("3") Then
            ReDim Horarios(5)
        Else
            ReDim Horarios(6)
        End If

        Dim resultadosPersistenciaHorariosTurno As Object =
PersistenciaHorarios.GetHorariosTurno(frm.lblValorTipoTurno.Tag)
        Dim readerHorariosTurno As MySqlDataReader =
resultadosPersistenciaHorariosTurno(0)

        Dim Row_Primera(8) As String
        Dim Row_Segunda(8) As String
        Dim Row_Tercera(8) As String
        Dim Row_Cuarta(8) As String
        Dim Row_Quinta(8) As String
        Dim Row_Sexta(8) As String
        Dim Row_Extra(8) As String

        Dim Rows As Object = {Row_Primera, Row_Segunda, Row_Tercera,
Row_Cuarta, Row_Quinta, Row_Sexta, Row_Extra}
```

```
        Dim WidgetDias As Object = {frm.Lunes, frm.Martes,
frm.Miércoles, frm.Jueves, frm.Viernes, frm.Sábado}

        Dim pos As Integer = 0
        While readerHorariosTurno.Read()
            Horarios(pos) =
readerHorariosTurno("HoraInicio").ToString()
            Rows(pos)(0) =
readerHorariosTurno("HoraInicio").ToString()
            Rows(pos)(0) = Rows(pos)(0).Substring(0,
Rows(pos)(0).Length - 3)
            pos += 1
        End While

        For Each Dia As String In Dias
            Dim resultadoPersistencia As Object =
PersistenciaHorarios.GetCalendarioDiarioForGrupo(Dia,
frm.cboGrupo.Text)
            Dim reader As MySqlDataReader = resultadoPersistencia(0)

            While reader.Read()
                Dim PosHora As Integer = Array.IndexOf(Horarios,
reader("HoraInicio").ToString())
                Dim PosDia As Integer = Array.IndexOf(Dias, Dia)
                Dim WidgetDia As Object = WidgetDias(PosDia)

                If reader("Materia").Equals("Sin asignar") Then
                    Rows(PosHora)(PosDia + 1) = "Sin definir" & vbCrLf
& ""
                    Continue While
                End If

                WidgetDia.agregarHora(reader("HoraOrden"),
reader("Materia"))
                Rows(PosHora)(PosDia + 1) = reader("Materia") & vbCrLf
& reader("NombreProfesor")
            End While

            reader.Close()
            resultadoPersistencia(1).Close()
        Next

        For Each row As Object In Rows
            pos = 0
            For Each Materia As String In row
                If String.IsNullOrWhiteSpace(Materia) Then
                    row(pos) = "Sin definir" & vbCrLf & ""
                End If
                pos += 1
            Next
        Next

        frm.Grilla.dgvMaterias.Rows.Add(Row_Primera)
        frm.Grilla.dgvMaterias.Rows.Add(Row_Segunda)
```

```
            frm.Grilla.dgvMaterias.Rows.Add(Row_Tercera)
            frm.Grilla.dgvMaterias.Rows.Add(Row_Cuarta)
            frm.Grilla.dgvMaterias.Rows.Add(Row_Quinta)
            frm.Grilla.dgvMaterias.Rows.Add(Row_Sexta)
            frm.Grilla.dgvMaterias.Rows.Add(Row_Extra)
            frm.Cursor = Cursors.Default

            readerHorariosTurno.Close()
            resultadosPersistenciaHorariosTurno(1).Close()
        End Sub
    End Class
```

9.1.4 Grupo.vb

```
Imports MySql.Data.MySqlClient

Public Class Grupo
    Public Shared Sub CargarGrupos(frm As frmAdminGrupos)
        ' Carga los grupos a la lista de grupos
        frm.pnlGrupos.Controls.Clear()
        frm.totalGrupos = 0
        frm.lblCantidadGrupos.Text = "(" + frm.totalGrupos.ToString()
+ ")"

        Dim primerGrupoFijado As Boolean = False
        Dim resultadosPersistencia As Object =
InformacionDB.GetGrupos()
        Dim reader As MySqlDataReader = resultadosPersistencia(0)

        While reader.Read()
            If frm.tipoUsuario.Equals("Adscripto") And
reader("CiPersona").Equals(frm.ciusuario) Then
                frm.AgregarWidgetGrupo(reader("NroGrupo"),
reader("Grado").ToString() & " " & reader("IdGrupo") &
ControlChars.NewLine & " (" & reader("NombreTurno") & ")",
reader("NombreTurno"), reader("CiPersona"))

                If Not primerGrupoFijado Then
                    frm.primerGrupo = reader("NroGrupo").ToString()
                    primerGrupoFijado = True
                End If
            ElseIf Not frm.tipoUsuario.Equals("Adscripto") Then
                frm.AgregarWidgetGrupo(reader("NroGrupo"),
reader("Grado").ToString() & " " & reader("IdGrupo") &
ControlChars.NewLine & " (" & reader("NombreTurno") & ")",
reader("NombreTurno"), reader("CiPersona"))
            End If
        End While
        reader.Close()

        resultadosPersistencia(1).Close()
    End Sub

    Public Shared Sub CargarSalones(frm As frmAdminGrupos)
        frm.cmbSalon.Items.Clear()
        frm.cmbSalon.Items.Add("Sin asignar")
        frm.cmbSalon.SelectedIndex = 0

        Dim resultadosPersistencia As Object =
InformacionDB.GetSalones()
        Dim reader As MySqlDataReader = resultadosPersistencia(0)

        While reader.Read()
            frm.cmbSalon.Items.Add(reader("IdSalon").ToString())
        End While
```

```
        reader.Close()

        resultadosPersistencia(1).Close()
    End Sub

    Public Shared Sub CargarGrados(frm As frmAdminGrupos)
        frm.cboGrado.Items.Clear()

        Try
            Dim resultadosPersistencia As Object =
InformacionDB.GetGrados(frm.cmbOrientacion.Text.Substring(0,
frm.cmbOrientacion.Text.IndexOf(" (")).Trim())
            Dim reader As MySqlDataReader = resultadosPersistencia(0)

            While reader.Read()
                frm.cboGrado.Items.Add(reader("Grado"))
            End While

            reader.Close()
            resultadosPersistencia(1).Close()
        Catch ex As Exception
        End Try
    End Sub

    Public Shared Sub CargarOrientaciones(frm As frmAdminGrupos)
        frm.cmbOrientacion.Items.Clear()
        Dim resultadosPersistencia =
InformacionDB.GetOrientaciones(frm.cmbCurso.Text.Substring(0,
frm.cmbCurso.Text.IndexOf(" (")).Trim())
        Dim reader As MySqlDataReader = resultadosPersistencia(0)

        While reader.Read()

frm.cmbOrientacion.Items.Add(reader("IdOrientacion").ToString() & " ("
& reader("NombreOrientacion").ToString() & ")")
        End While
        reader.Close()

        resultadosPersistencia(1).Close()
    End Sub

    Public Shared Sub CargarCursos(frm As frmAdminGrupos)
        frm.cmbCurso.Items.Clear()

        Dim resultadosPersistencia = InformacionDB.GetCursos()
        Dim reader As MySqlDataReader = resultadosPersistencia(0)

        While reader.Read()
            frm.cmbCurso.Items.Add(reader("IdCurso").ToString() & " ("
& reader("NombreCurso") & ")")
        End While
        reader.Close()

        resultadosPersistencia(1).Close()
```

```
    End Sub

    Public Shared Sub CargarTurnos(frm As frmAdminGrupos)
        frm.cmbTurno.Items.Clear()

        Dim resultadosPersistencia = InformacionDB.GetTurnos()
        Dim reader As MySqlDataReader = resultadosPersistencia(0)

        While reader.Read()
            frm.cmbTurno.Items.Add(reader("NombreTurno"))
        End While
        reader.Close()

        resultadosPersistencia(1).Close()
    End Sub

    Public Shared Sub CargarGrupo(nroGrupo As String, frm As
frmAdminGrupos)
        Dim resultadosPersistencia As Object =
PersistenciaGrupos.GetInfo(nroGrupo)
        Dim reader As MySqlDataReader = resultadosPersistencia(0)

        While reader.Read()
            frm.txtIdGrupo.Text = reader("IdGrupo")
            frm.cmbCurso.SelectedIndex =
frm.cmbCurso.FindStringExact(reader("IDCurso").ToString() & " (" &
reader("NombreCurso") & ")")
            frm.chkDiscapacitado.Checked = reader("Discapacitado")
            frm.cmbTurno.SelectedIndex = reader("IDTurno") - 1

            frm.cmbOrientacion.Items.Clear()

frm.cmbOrientacion.Items.Add(reader("IdOrientacion").ToString() & " ("
& reader("NombreOrientacion").ToString() & ")")
            frm.cmbOrientacion.SelectedIndex = 0

            frm.cboGrado.Items.Clear()
            frm.cboGrado.Items.Add(reader("Grado"))
            frm.cboGrado.SelectedIndex = 0

            frm.cmbAdscripto.SelectedIndex =
frm.cmbAdscripto.FindStringExact(reader("NombreAdscripto"))
            If reader("NombreAdscripto").Equals("-1 - Sin definir")
Then
                frm.cmbAdscripto.SelectedIndex = 0
            End If

            CargarSalones(frm)
            Dim salon As String = reader("IdSalon")
            If salon.Equals("-1") Then
                salon = "Sin asignar"
            End If
            frm.cmbSalon.SelectedIndex =
frm.cmbSalon.FindStringExact(salon)
```

```
            frm.cmbOrientacion.Enabled = False
            frm.cboGrado.Enabled = False
            frm.cmbAdscripto.Enabled = False
        End While

        reader.Close()
        resultadosPersistencia(1).Close()
    End Sub

    Public Shared Sub CargarAdscriptos(frm As frmAdminGrupos)
        frm.cmbAdscripto.Items.Clear()

        Dim resultadosPersistencia As Object =
InformacionDB.GetAdscriptos()
        Dim reader As MySqlDataReader = resultadosPersistencia(0)

        While reader.Read()
            If reader("CiPersona").Equals("-1") Then
                frm.cmbAdscripto.Items.Add("Sin definir")
                Continue While
            End If
            frm.cmbAdscripto.Items.Add(reader("CiPersona") & " - " &
reader("Adscripto"))
        End While

        frm.cmbAdscripto.SelectedIndex = 0

        reader.Close()
        resultadosPersistencia(1).Close()
    End Sub

    Public Shared Sub CheckSalonOcupado(frm As frmAdminGrupos)

        Dim salon As String
        salon = frm.cmbSalon.Text
        If salon.Equals("Sin asignar") Then
            salon = "-1"
        End If

        Dim resultadosPersistencia As Object =
PersistenciaSalones.GetOcupadoBy(salon, frm.cmbTurno.SelectedIndex +
1)
        Dim reader As MySqlDataReader = resultadosPersistencia(0)

        While reader.Read()
            If reader("IdGrupo").Equals(frm.txtIdGrupo.Text) And
reader("Grado").Equals(Integer.Parse(frm.cboGrado.Text)) Then
                Continue While
            End If

            reader.Close()
            resultadosPersistencia(1).Close()
```

```
            Throw New System.Exception("Duplicate")
        End While

        reader.Close()
        resultadosPersistencia(1).Close()
    End Sub

    Public Shared Sub ActualizarDB(frm As frmAdminGrupos)
        ' Agrega un salón a la base de datos
        Dim IdGrupo As String = frm.txtIdGrupo.Text
        Dim Discapacitado As Boolean = frm.chkDiscapacitado.Checked
        Dim IdSalon As String = frm.cmbSalon.Text
        Dim IdTurno As String = frm.cmbTurno.SelectedIndex + 1
        Dim Grado As String = frm.cboGrado.Text
        Dim IdOrientacion As String =
frm.cmbOrientacion.Text.Substring(0, frm.cmbOrientacion.Text.IndexOf("
(")).Trim()
        Dim CiAdscripto As String

        If IdSalon.Equals("Sin asignar") Then
            IdSalon = "-1"
        End If

        If Not frm.cmbAdscripto.Text.Equals("Sin definir") Then
            CiAdscripto = frm.cmbAdscripto.Text.Substring(0,
frm.cmbAdscripto.Text.IndexOf(" - ")).Trim()
        Else
            CiAdscripto = "-1"
        End If

        Try
            CheckSalonOcupado(frm)
        Catch ex As Exception
            MessageBox.Show("El salón escogido ya está en uso.",
"Salón en uso", MessageBoxButtons.OK, MessageBoxIcon.Stop)
            Return
        End Try

        If frm.btnAgregar.Text.Equals("Agregar grupo") Then
            Try
                If PersistenciaGrupos.GetExiste(frm.cboGrado.Text & "
" & frm.txtIdGrupo.Text) Then
                    Throw New System.Exception("Duplicate")
                End If

                PersistenciaGrupos.Add(IdGrupo, Discapacitado,
IdSalon, IdTurno, Grado, IdOrientacion, CiAdscripto)
            Catch ex As Exception
                If ex.ToString().Contains("Duplicate") Then
                    MessageBox.Show("Ya existe un grupo con ese grado
e id.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
                Else
                    MessageBox.Show(ex.ToString(), "Error",
MessageBoxButtons.OK, MessageBoxIcon.Warning)
```

```
                End If
                Return
            End Try

            If frm.chkDistribuir.Checked Then
                frm.frmAdministrar.habilitarBotones(False)
                frm.ParentForm.Enabled = False

                Dim ventanaEspere As New frmDialogoEspere()
                ventanaEspere.Show()
                ventanaEspere.lblComprobando.Text = "Repartiendo
horarios"

                Dim Magia As New Magia()
                Magia.RepartirHorarios(frm)

                frm.ParentForm.Enabled = True
                frm.frmAdministrar.habilitarBotones(True)
                ventanaEspere.Dispose()
            End If

            CargarGrupos(frm)
            MessageBox.Show("Grupo agregado correctamente", "Grupo
agregado", MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
        Else
            PersistenciaGrupos.Edit(IdGrupo, Discapacitado, IdSalon,
IdTurno, Grado, CiAdscripto)
            MessageBox.Show("Datos del grupos actualizados
correctamente", "Grupo actualizado", MessageBoxButtons.OK,
MessageBoxIcon.Asterisk)
        End If

        frm.InterfazNuevoGrupo()
        frm.btnNuevoGrupo.Text = "Agregar grupo"
        frm.frmMain.RecargarGrupo()

    End Sub

    Public Shared Sub EliminarGrupo(nroGrupo As String, nombreGrupo As
String, frm As frmAdminGrupos)
        Dim backupMaterias As Object =
PersistenciaHorarios.GetForGrupo(nroGrupo)

        Try
            PersistenciaHorarios.DelGrupoEntero(nroGrupo)
            PersistenciaGrupos.Del(nroGrupo)
            CargarGrupos(frm)
            frm.InterfazNuevoGrupo()

            MessageBox.Show("Grupo '" + nombreGrupo + "' eliminado.",
"Grupo eliminado.", MessageBoxButtons.OK, MessageBoxIcon.Information)
        Catch ex As Exception
            Dim reader As MySqlDataReader = backupMaterias(0)
```

```
            Try
                While reader.Read()
                    PersistenciaHorarios.Add(reader("IdAsignatura"),
reader("NroGrupo"), reader("HoraInicio"), reader("HoraFin"),
reader("Dia"), reader("IdTurno"), reader("CiPersona"))
                End While
            Catch exx As Exception
            End Try

            MessageBox.Show("No se pudo eliminar el grupo, el mismo
tiene docentes (ver admin. de docentes) asignados.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try

        backupMaterias(0).Close()
        backupMaterias(1).Close()
    End Sub
End Class
```

9.1.5 InformacionDB.vb

```
Imports MySql.Data.MySqlClient
Imports System.Data

Public Class InformacionDB
    Public Shared Function GetSalones() As Object
        Dim conexion As New Conexion()

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "SELECT * FROM `Salon` where not
IdSalon='-1';"
                .CommandType = CommandType.Text
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function

    Public Shared Function GetGrupos(Optional IdCurso As String =
Nothing, Optional IdTurno As String = Nothing) As Object
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                If Not IsNothing(IdCurso) And Not IsNothing(IdTurno)
Then
                    .CommandText = "SELECT Grupo.*, Turno.NombreTurno
from `Grupo`, `Orientacion`, `Turno` where Grupo.IdTurno=@IdTurno and
Grupo.IdTurno=Turno.IdTurno and
Orientacion.IdOrientacion=Grupo.IdOrientacion and
Orientacion.IdCurso=@IdCurso;"
                    .Parameters.AddWithValue("@IdCurso", IdCurso)
                    .Parameters.AddWithValue("@IdTurno", IdTurno)
                ElseIf Not IsNothing(IdCurso) Then
                    .CommandText = "SELECT Grupo.*, Turno.NombreTurno
from `Grupo`, `Orientacion`, `Turno` where Grupo.IdTurno=Turno.IdTurno
and Orientacion.IdOrientacion=Grupo.IdOrientacion and
Orientacion.IdCurso=@IdCurso;"
                    .Parameters.AddWithValue("@IdCurso", IdCurso)
                ElseIf Not IsNothing(IdTurno) Then
                    .CommandText = "SELECT Grupo.*, Turno.NombreTurno
from `Grupo`, `Turno` where Grupo.IdTurno=@IdTurno and
Grupo.IdTurno=Turno.IdTurno;"
                    .Parameters.AddWithValue("@IdTurno", IdTurno)
                Else
                    .CommandText = "SELECT Grupo.*, Turno.NombreTurno
FROM `Grupo`, `Turno` WHERE Grupo.IdTurno=Turno.IdTurno;"
                End If
```

```
                .CommandType = CommandType.Text
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function

    Public Shared Function GetCursos() As Object
        Dim conexion As New Conexion()

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "SELECT * FROM `Curso`;"
                .CommandType = CommandType.Text
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function

    Public Shared Function GetOrientaciones(IdCurso As String) As
Object
        Dim conexion As New Conexion()

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "SELECT * FROM `Orientacion` WHERE
IdCurso=@IdCurso;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@IdCurso", IdCurso)
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function

    Public Shared Function GetGrados(IdOrientacion As String) As
Object
        Dim conexion As New Conexion()

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "select * from Trayecto where
IdOrientacion=@IdOrientacion;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@IdOrientacion",
IdOrientacion)
            End With
```

```
            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function

    Public Shared Function GetTurnos() As Object
        Dim conexion As New Conexion()

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "SELECT * FROM `Turno`;"
                .CommandType = CommandType.Text
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function

    Public Shared Function GetAdscriptos() As Object
        Dim conexion As New Conexion()

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "select * from Adscriptos;"
                .CommandType = CommandType.Text
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function

    Public Shared Function GetUsuarios() As Object
        Dim conexion As New Conexion()

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "SELECT * FROM `Usuario`;"
                .CommandType = CommandType.Text
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function

    Public Shared Function GetAreasOrientacionByGrado(IdOrientacion As
String, Grado As String) As Object
        Dim conexion As New Conexion()
```

```
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "select DISTINCT Area.IdArea,
NombreArea from (select Asignatura.IdArea from Tiene_Ta, Asignatura
where Tiene_Ta.IdAsignatura=Asignatura.IdAsignatura and
Tiene_Ta.IdOrientacion=@IdOrientacion and Tiene_Ta.Grado=@Grado)
Orientacion, Area where Orientacion.IdArea=Area.IdArea;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@IdOrientacion",
IdOrientacion)
                .Parameters.AddWithValue("@Grado", Grado)
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function

    Public Shared Function GetDocentes() As Object
        Dim conexion As New Conexion()

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "SELECT Profesor.CiPersona,
Persona.NombrePersona, Persona.ApellidoPersona FROM `Profesor`,
`Persona` where Profesor.CiPersona=Persona.CiPersona;"
                .CommandType = CommandType.Text
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function

    Public Shared Function GetAsignaturasForGrupo(IdGrupo As String,
Grado As String, IdArea As String) As Object
        Dim conexion As New Conexion()

        Dim IdOrientacion As String =
PersistenciaGrupos.GetOrientacion(IdGrupo, Grado)

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "select Asignatura.* from (select
DISTINCT Area.IdArea, NombreArea from (select Asignatura.IdArea from
Tiene_Ta, Asignatura where
Tiene_Ta.IdAsignatura=Asignatura.IdAsignatura and
Tiene_Ta.IdOrientacion=@IdOrientacion) Orientacion, Area where
Orientacion.IdArea=Area.IdArea) Areas, Asignatura, Tiene_Ta where
Asignatura.IdArea=Areas.IdArea and
```

```
Tiene_Ta.IdAsignatura=Asignatura.IdAsignatura and
Tiene_Ta.Grado=@Grado and IdOrientacion=@IdOrientacion and
Asignatura.IdArea=@IdArea;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@Grado", Grado)
                .Parameters.AddWithValue("@IdOrientacion",
IdOrientacion)
                .Parameters.AddWithValue("@IdArea", IdArea)
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function

    Public Shared Function GetAsignaturasReparticion(IdGrupo As
String, Grado As String) As Object
        Dim conexion As New Conexion()

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "select T.* from Tiene_Ta T, Grupo G,
Asignatura A where T.IdAsignatura=A.IdAsignatura and
T.IdOrientacion=G.IdOrientacion and G.IdGrupo=@IdGrupo and
G.Grado=@Grado and T.Grado=G.Grado order by `CargaHoraria` DESC;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@IdGrupo", IdGrupo)
                .Parameters.AddWithValue("@Grado", Grado)
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function
End Class
```

9.1.6 Magia.vb

```
Imports MySql.Data.MySqlClient

Public Class Magia
    Dim max_horas_diarias As Integer = 7
    Dim materias_auxiliares(1) As Array
    Dim materias_ordenadas(1) As Array
    Dim asignacionLunes(1) As String
    Dim asignacionMartes(1) As String
    Dim asignacionMiercoles(1) As String
    Dim asignacionJueves(1) As String
    Dim asignacionViernes(1) As String
    Dim asignacionSabado(1) As String

    Private Sub CargarMateriasAsignacion(frm As frmAdminGrupos)
        Dim resultadosPersistencia As Object =
InformacionDB.GetAsignaturasReparticion(frm.txtIdGrupo.Text,
frm.cboGrado.Text)
        Dim reader As MySqlDataReader = resultadosPersistencia(0)
        Dim pos As Integer = 0
        While reader.Read()
            materias_auxiliares(pos) = {reader("IdAsignatura"),
reader("CargaHoraria"), reader("EnseniaDeCorrido")}
            pos += 1
            ReDim Preserve materias_auxiliares(pos + 1)
        End While
        reader.Close()

        ReDim Preserve materias_auxiliares(pos - 1)
    End Sub

    Private Sub CrearModulos()
        Dim pos As Integer = 0

        For Each materia As Object In materias_auxiliares
            If Not materia(2) And materia(1) > 1 Then
                While materia(1) > 0
                    If materia(1) = 1 Then
                        materia = {materia(0), 0, False}
                        materias_ordenadas(pos) = {materia(0), 1, "-"}

                    ElseIf materia(1) = 2 Then
                        materia = {materia(0), 0, False}
                        materias_ordenadas(pos) = {materia(0), 2, "-"}

                    Else
                        materia = {materia(0), materia(1) - 2, False}
                        materias_ordenadas(pos) = {materia(0), 2, "-"}
                    End If

                    pos += 1
                    ReDim Preserve materias_ordenadas(pos + 1)
```

```
                End While
            ElseIf materia(2) Then
                materias_ordenadas(pos) = {materia(0), materia(1), "--
"}
                pos += 1

                ReDim Preserve materias_ordenadas(pos + 1)
            End If
        Next

        ReDim Preserve materias_ordenadas(materias_ordenadas.Length -
1)
    End Sub

    Public Sub RepartirHorarios(frm As frmAdminGrupos)
        Try
            RepartirHorariosAutomagicamente(frm)
        Catch ex As Exception
            MessageBox.Show("Hubieron errores al hacer la asignación
inicial de materias." & vbCrLf & "Deberá modificar las asignaturas
manualmente en la pestaña horarios", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Warning)
        End Try
    End Sub

    Private Sub RepartirHorariosAutomagicamente(frm As frmAdminGrupos)
        Dim NroGrupo As String =
PersistenciaGrupos.GetNroGrupo(frm.cboGrado.Text + " " +
frm.txtIdGrupo.Text)
        Dim IdTurno As String =
PersistenciaGrupos.GetTurno(frm.cboGrado.Text + " " +
frm.txtIdGrupo.Text)

        CargarMateriasAsignacion(frm)
        CrearModulos()

        Dim materias_asignadas(materias_ordenadas.Length) As Array

        Dim posLunes As Integer = 0
        Dim posMartes As Integer = 0
        Dim posMiercoles As Integer = 0
        Dim posJueves As Integer = 0
        Dim posViernes As Integer = 0
        Dim posSabado As Integer = 0
        Dim pos As Integer = 0

        If IdTurno = 1 Then
            posLunes = 1
            posMartes = 1
            posMiercoles = 1
            posJueves = 1
            posViernes = 1
            posSabado = 1
        End If
```

```
        For value As Integer = 0 To 50
            For Each materia As Object In materias_ordenadas
                If materias_asignadas.Contains(materia) Then
                    Continue For
                End If

                If value > 25 And Not IdTurno = 3 Then
                    max_horas_diarias = 8
                Else
                    max_horas_diarias = 7
                End If


                If max_horas_diarias > asignacionLunes.Length - 1 +
materia(1) And Not asignacionLunes.Contains(materia(0)) Then
                    Dim x As Integer = 0
                    While x < materia(1)
                        asignacionLunes(posLunes) = materia(0)
                        x += 1
                        posLunes += 1
                        ReDim Preserve asignacionLunes(posLunes)
                    End While

                    materias_asignadas(pos) = materia
                    pos += 1

                ElseIf max_horas_diarias > asignacionMartes.Length - 1
+ materia(1) And Not asignacionMartes.Contains(materia(0)) Then
                    Dim x As Integer = 0
                    While x < materia(1)
                        asignacionMartes(posMartes) = materia(0)
                        x += 1
                        posMartes += 1
                        ReDim Preserve asignacionMartes(posMartes)
                    End While

                    materias_asignadas(pos) = materia
                    pos += 1

                ElseIf max_horas_diarias > asignacionMiercoles.Length
- 1 + materia(1) And Not asignacionMiercoles.Contains(materia(0)) Then
                    Dim x As Integer = 0
                    While x < materia(1)
                        asignacionMiercoles(posMiercoles) = materia(0)
                        x += 1
                        posMiercoles += 1
                        ReDim Preserve
asignacionMiercoles(posMiercoles)
                    End While

                    materias_asignadas(pos) = materia
                    pos += 1
```

```
                ElseIf max_horas_diarias > asignacionJueves.Length - 1
+ materia(1) And Not asignacionJueves.Contains(materia(0)) Then
                    Dim x As Integer = 0
                    While x < materia(1)
                        asignacionJueves(posJueves) = materia(0)
                        x += 1
                        posJueves += 1
                        ReDim Preserve asignacionJueves(posJueves)
                    End While

                    materias_asignadas(pos) = materia
                    pos += 1

                ElseIf max_horas_diarias > asignacionViernes.Length -
1 + materia(1) And Not asignacionViernes.Contains(materia(0)) Then
                    Dim x As Integer = 0
                    While x < materia(1)
                        asignacionViernes(posViernes) = materia(0)
                        x += 1
                        posViernes += 1
                        ReDim Preserve asignacionViernes(posViernes)
                    End While

                    materias_asignadas(pos) = materia
                    pos += 1

                ElseIf max_horas_diarias > asignacionSabado.Length - 1
+ materia(1) And Not asignacionSabado.Contains(materia(0)) And
max_horas_diarias <= 7 Then
                    Dim x As Integer = 0
                    While x < materia(1)
                        asignacionSabado(posSabado) = materia(0)
                        x += 1
                        posSabado += 1
                        ReDim Preserve asignacionSabado(posSabado)
                    End While

                    materias_asignadas(pos) = materia
                    pos += 1

                Else
                End If
            Next
        Next

        ReDim Preserve asignacionLunes(asignacionLunes.Length - 2)
        ReDim Preserve asignacionMartes(asignacionMartes.Length - 2)
        ReDim Preserve asignacionMiercoles(asignacionMiercoles.Length
- 2)
        ReDim Preserve asignacionJueves(asignacionJueves.Length - 2)
        ReDim Preserve asignacionViernes(asignacionViernes.Length - 2)
        ReDim Preserve asignacionSabado(asignacionSabado.Length - 2)
```

```
        Dim HorariosInicio(8) As Object
        Dim HorariosFin(8) As Object

        Dim resultadosPersistencia As Object =
PersistenciaHorarios.GetHorariosTurno(IdTurno)
        Dim reader As MySqlDataReader = resultadosPersistencia(0)
        Dim posActual As Integer = 1

        While reader.Read()
            HorariosInicio(posActual) =
reader("HoraInicio").ToString()
            HorariosFin(posActual) = reader("HoraFin").ToString()

            posActual += 1
        End While

        reader.Close()
        resultadosPersistencia(1).Close()

        Dim total As Integer = 0
        Dim horaActual As Integer = 1
        Dim sentencias(0) As Array
        Dim Dias As String() = {"Lunes", "Martes", "Miércoles",
"Jueves", "Viernes", "Sábado"}
        Dim Asignacion As Object = {asignacionLunes, asignacionMartes,
asignacionMiercoles, asignacionJueves, asignacionViernes,
asignacionSabado}

        Dim DiaActual As Integer = 0
        For Each asigna As Object In Asignacion
            horaActual = 1
            For Each item As String In asigna
                Dim Dia As String = Dias(DiaActual)
                sentencias(total) = {item, NroGrupo,
HorariosInicio(horaActual), HorariosFin(horaActual), Dia, IdTurno, "-
1"}

                horaActual += 1
                total += 1
                ReDim Preserve sentencias(total)
            Next
            DiaActual += 1
        Next

        ReDim Preserve sentencias(sentencias.Length - 1)
        Dim huboError = False

        For Each sentencia As Object In sentencias
            Try
                PersistenciaHorarios.Add(sentencia(0), sentencia(1),
sentencia(2), sentencia(3), sentencia(4), sentencia(5), sentencia(6))
            Catch ex As Exception
            End Try
        Next
```

```
        For Each materia As Object In materias_ordenadas
            If materias_asignadas.Contains(materia) Then
                Continue For
            End If
            Try
                Dim IdAsignatura As String = materia(0)
                Throw New System.Exception("No se pueden asignar :(")
            Catch ex As Exception
                If ex.ToString().Contains("No se pueden") Then
                    Throw New System.Exception("No se pueden asignar
:(")
                End If
            End Try
            Exit For
        Next

    End Sub
End Class
```

9.1.7 Salon.vb

```
Imports MySql.Data.MySqlClient

Public Class Salon

    Public Shared Sub CargarSalones(frm As frmAdminSalones)
        ' Carga los salones y los pone en la lista
        frm.pnlSalones.Controls.Clear()
        frm.totalSalones = 0
        frm.lblCantidadSalones.Text = "(" +
frm.totalSalones.ToString() + ")"

        Dim primerSalonFijado As Boolean = False

        Dim resultadosPersistencia As Object =
InformacionDB.GetSalones()
        Dim reader As MySqlDataReader = resultadosPersistencia(0)

        While reader.Read()
            frm.agregarWidgetSalon(reader("IdSalon"))

            If Not primerSalonFijado Then
                frm.primerSalon = reader("IdSalon")
                primerSalonFijado = True
            End If
        End While

        reader.Close()
        resultadosPersistencia(1).Close()
    End Sub

    Public Shared Sub ActualizarDB(frm As frmAdminSalones)
        ' Agrega o actualiza los datos del salón en la DB
        If frm.btnAgregar.Text.Equals("Agregar salón") Then
            Try
                PersistenciaSalones.Add(frm.txtIdSalon.Text,
frm.txtComentarios.Text, frm.cmbPlanta.Text)
                MessageBox.Show("Salón agregado correctamente", "Salón
agregado", MessageBoxButtons.OK, MessageBoxIcon.Asterisk)

                frm.CargarSalones()
                frm.InterfazPrevisualizarSalon(frm.txtIdSalon.Text)
            Catch ex As Exception
                MessageBox.Show("Ya existe un salón con ese ID.",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            End Try
        Else
            PersistenciaSalones.Edit(frm.txtIdSalon.Text,
frm.txtComentarios.Text, frm.cmbPlanta.Text)
            MessageBox.Show("Información de salón actualizada
correctamente", "Salón actualizado", MessageBoxButtons.OK,
MessageBoxIcon.Asterisk)
```

```
            frm.CargarSalones()
            frm.InterfazPrevisualizarSalon(frm.txtIdSalon.Text)
        End If
    End Sub

    Public Shared Sub CargarSalon(IdSalon As String, frm As
frmAdminSalones)
        '  Carga los datos de un salón
        Dim resultadoPersistencia As Object =
PersistenciaSalones.GetInfo(IdSalon)
        Dim reader As MySqlDataReader = resultadoPersistencia(0)

        While reader.Read()
            frm.txtIdSalon.Text = reader("IdSalon")
            frm.txtComentarios.Text =
reader("ComentariosSalon").ToString()
            frm.cmbPlanta.SelectedIndex =
frm.cmbPlanta.FindStringExact(reader("PlantaSalon"))
        End While

        reader.Close()
        resultadoPersistencia(1).Close()

        ' Carga los horarios del salón.

        For Turno As Integer = 1 To 3
            resultadoPersistencia =
PersistenciaSalones.GetAsignado(IdSalon, Turno)
            reader = resultadoPersistencia(0)
            While reader.Read()
                If Turno = 1 Then
                    frm.lblSalonMatutino.Text = reader("Grupo")
                ElseIf Turno = 2 Then
                    frm.lblSalonVespertino.Text = reader("Grupo")
                ElseIf Turno = 3 Then
                    frm.lblSalonNocturno.Text = reader("Grupo")
                End If
            End While
            reader.Close()
            resultadoPersistencia(1).Close()
        Next
    End Sub

    Public Shared Sub EliminarSalon(IdSalon As String, frm As
frmAdminSalones)
        Try
            PersistenciaSalones.Del(IdSalon)
            frm.cargarSalones()
            frm.IntefazNuevoSalon()
            MessageBox.Show("Salón '" + (IdSalon) + "' eliminado.",
"Salón eliminado.", MessageBoxButtons.OK, MessageBoxIcon.Information)
        Catch ex As Exception
```

```
            MessageBox.Show("El salón no se puede eliminar, ya que
está asignado a un grupo.", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
        End Try
    End Sub
End Class
```

9.1.8 Usuario.vb

```
Imports MySql.Data.MySqlClient

Public Class Usuario
    Public Shared Sub Login(frm As frmLogin)
        ' Se encarga de comprobar los datos ingresados del usuario,
con los de la DB
        Dim accesoDenegado As Boolean = True
        Dim conexion As New Conexion()
        Dim tipoUsuario As String = ""

        Dim resultadosPersistencia As Object =
PersistenciaUsuarios.Login(frm.txtCi.Text, frm.txtContraseña.Text)
        Dim reader As MySqlDataReader = resultadosPersistencia(0)
        While reader.Read()
            If Not reader("AprobacionUsuario") Then
                MsgBox("Acceso no autorizado" & vbCrLf & "El
administrador aún debe confirmar su registro",
MsgBoxStyle.Information, "Minerva · Registro a confirmar")
                frm.lblDatosInc.Text = "Cuenta no autorizada"
                frm.pnlError.Visible = True
                frm.Cursor = Cursors.Default
                Return
            End If
            tipoUsuario = reader("TipoUsuario")
            accesoDenegado = False
        End While
        reader.Close()
        resultadosPersistencia(1).Close()

        frm.Cursor = Cursors.Default
        If accesoDenegado Then
            frm.lblDatosInc.Text = "Datos incorrectos!"
            frm.pnlError.Visible = True
        Else
            Dim minerva As New frmMain(False, frm.txtCi.Text,
tipoUsuario)

            minerva.Show()
            minerva.BringToFront()
            frm.Hide()
        End If
    End Sub

    Public Shared Sub Registro(frm As frmRegistro)
        Dim cantidadAdministradores As Integer =
PersistenciaUsuarios.CantAdministradores()

        Try
            PersistenciaPersonas.Add(frm.txtCi.Text)
```

```
            Dim UsuarioAprobado As Boolean = False
            Dim TipoUsuario As String = ""

            If cantidadAdministradores <= 0 Then
                UsuarioAprobado = True
                TipoUsuario = "Administrador"
            Else
                If frm.radFuncionario.Checked Then
                    TipoUsuario = "Funcionario"

                ElseIf frm.radAdscripto.Checked Then
                    TipoUsuario = "Adscripto"

                ElseIf frm.radAdministrador.Checked Then
                    TipoUsuario = "Administrador"
                End If

                UsuarioAprobado = False
            End If

            PersistenciaUsuarios.Add(frm.txtCi.Text, TipoUsuario,
frm.txtContraseña.Text, UsuarioAprobado)

            If cantidadAdministradores <= 0 Then
                MsgBox("Gracias por registrarse. " & vbCrLf & "Usted
ha sido registrado automáticamente como administrador." & vbCrLf & "Ya
puede acceder al sistema utilizando sus datos.",
MsgBoxStyle.Information, "Minerva · Confirmación de registro")
            Else
                MsgBox("Gracias por registrarse. " & vbCrLf & "El
administrador deberá confirmar su registro", MsgBoxStyle.Information,
"Minerva · Confirmación de registro")
            End If

            frm.Hide()
            frmIngresarRegistro.Show()
            frmIngresarRegistro.BringToFront()
            frm.Dispose()
        Catch ex As Exception
            frm.pnlError.Visible = True
        End Try
    End Sub

    Public Shared Sub EditarDatos(frm As frmDatosUsuario)
        PersistenciaPersonas.Edit(frm.frmMain.nombreUsuario,
frm.txtNombre.Text, frm.txtApellido.Text)
        MessageBox.Show("Información de usuario actualizada
correctamente", "Usuario actualizado", MessageBoxButtons.OK,
MessageBoxIcon.Asterisk)
        frm.frmMain.BringToFront()
        frm.Dispose()
    End Sub
```

```
    Public Shared Sub CargarUsuarios(frm As frmAdminUsuarios)
        frm.pnlUsuarios.Controls.Clear()
        frm.totalUsuarios = 0

        Dim resultadosPersistencia As Object =
InformacionDB.GetUsuarios()
        Dim reader As MySqlDataReader = resultadosPersistencia(0)
        While reader.Read()
            If reader("CiPersona").Equals("-1") Then
                Continue While
            End If
            frm.AgregarWidgetUsuario(reader("CiPersona"),
reader("TipoUsuario"), reader("AprobacionUsuario"))
        End While
        reader.Close()
        resultadosPersistencia(1).Close()
    End Sub

    Public Shared Sub EliminarUsuario(Ci As String, frm As
frmAdminUsuarios)
        Dim resultadosPersistenciaBackup As Object =
PersistenciaUsuarios.GetInfo(Ci)
        Dim readerBackup As MySqlDataReader =
resultadosPersistenciaBackup(0)

        Try
            PersistenciaUsuarios.Del(Ci)
            PersistenciaPersonas.Del(Ci)
            MessageBox.Show("Usuario '" + Ci + "' eliminado.",
"Usuario eliminado.", MessageBoxButtons.OK,
MessageBoxIcon.Information)
            CargarUsuarios(frm)
            frm.InterfazNuevoUsuario()
        Catch ex As Exception
            While readerBackup.Read()
                PersistenciaUsuarios.Add(readerBackup("CiPersona"),
readerBackup("TipoUsuario"), readerBackup("ContraseniaUsuario"),
readerBackup("AprobacionUsuario"))
            End While
            MessageBox.Show("El adscripto está asignado a un grupo, no
se puede eliminar.", "Error al eliminar adscripto",
MessageBoxButtons.OK, MessageBoxIcon.Stop)
        End Try

        readerBackup.Close()
        resultadosPersistenciaBackup(1).Close()
    End Sub

    Public Shared Sub CargarDatos(Ci As String, frm As
frmAdminUsuarios)
        Dim resultadosPersistencia As Object =
PersistenciaUsuarios.GetInfo(Ci)
        Dim reader As MySqlDataReader = resultadosPersistencia(0)
```

```
        While reader.Read()
            frm.txtID.Text = reader("CiPersona")
            frm.txtContraseña.Text = reader("ContraseniaUsuario")
            frm.chkHabilitado.Checked = reader("AprobacionUsuario")
            frm.tipoSeleccionado = reader("TipoUsuario")
            Try
                frm.txtNombre.Text = reader("NombrePersona")
                frm.txtApellido.Text = reader("ApellidoPersona")
            Catch ex As Exception
                frm.txtNombre.Text = ""
                frm.txtApellido.Text = ""
            End Try
            If reader("TipoUsuario").Equals("Funcionario") Then
                frm.radFuncionario.Checked = True
            ElseIf reader("TipoUsuario").Equals("Administrador") Then
                frm.radAdministrador.Checked = True
            ElseIf reader("TipoUsuario").Equals("Adscripto") Then
                frm.radAdscripto.Checked = True
            End If
        End While

        reader.Close()
        resultadosPersistencia(1).Close()
    End Sub

    Public Shared Sub ActualizarDB(frm As frmAdminUsuarios)
        Dim CiPersona As String = frm.txtID.Text
        Dim ContraseniaUsuario As String = frm.txtContraseña.Text
        Dim AprobacionUsuario As Boolean = frm.chkHabilitado.Checked
        Dim Tipousuario As String = frm.tipoSeleccionado
        Dim NombrePersona As String = frm.txtNombre.Text
        Dim ApellidoPersona As String = frm.txtApellido.Text

        If frm.btnAgregar.Text.Equals("Agregar usuario") Then
            Try
                PersistenciaPersonas.Add(CiPersona, NombrePersona,
ApellidoPersona)
                PersistenciaUsuarios.Add(CiPersona, Tipousuario,
ContraseniaUsuario, AprobacionUsuario)
                MessageBox.Show("Usuario agregado correctamente",
"Usuario agregado", MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
                CargarUsuarios(frm)
                frm.InterfazNuevoUsuario()
            Catch ex As Exception
                If ex.ToString().Contains("Duplicate") Then
                    MessageBox.Show("Ya existe un usuario (o
profesor!) con ese ID.", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
                Else
                    MessageBox.Show(ex.ToString(), "Error",
MessageBoxButtons.OK, MessageBoxIcon.Warning)
                End If
            End Try
        Else
```

```
            PersistenciaPersonas.Edit(CiPersona, NombrePersona,
ApellidoPersona)
            PersistenciaUsuarios.Edit(CiPersona, Tipousuario,
ContraseniaUsuario, AprobacionUsuario)
            MessageBox.Show("Información de usuario actualizada
correctamente", "Usuario actualizado", MessageBoxButtons.OK,
MessageBoxIcon.Asterisk)
            CargarUsuarios(frm)
            frm.InterfazNuevoUsuario()
        End If
    End Sub

    Public Shared Sub ContUsuariosNoAprobados(frm As frmMain)
        If Not frm.tipoUsuario.Equals("Administrador") Then
            frm.alertaAprobacion.Visible = False
            Return
        End If

        Dim cantidadAprobar As Integer =
PersistenciaUsuarios.CantSinAprobar()
        Dim notificaciones As Object = {notificacion_1,
notificacion_2, notificacion_3, notificacion_4, notificacion_5,
notificacion_6, notificacion_7, notificacion_8, notificacion_9,
notificacion_10, notificacion_max}
        If cantidadAprobar = 1 Then
            frm.lblCantidadUsuariosAprobacion.Text = "1 usuario está
esperando la aprobación de un administrador para poder ingresar."
        Else
            frm.lblCantidadUsuariosAprobacion.Text = cantidadAprobar &
" usuarios están esperando la aprobación de un administrador para
poder ingresar."
        End If
        If cantidadAprobar <= 0 Then
            frm.alertaAprobacion.Visible = False
        Else
            frm.alertaAprobacion.Visible = True
            If cantidadAprobar > 10 Then
                cantidadAprobar = 11
            End If
            frm.alertaAprobacion.BackgroundImage =
notificaciones(cantidadAprobar - 1)
        End If
    End Sub
End Class
```

9.2 Capa de lógica

9.2.1 Conexión.vb

```
Imports MySql.Data.MySqlClient ' Importa el módulo de MySQL

Public Class Conexion
    ' Define la conexión como variable pública
    Friend Conn As MySqlConnection
    Dim server, user, passwd, db As String

    Public Sub New(Optional noSalir As Boolean = False)
        ' Al crear la clase, generar la conexión, y abrirla
        Try
            server = GetSetting("Minerva", "BaseDeDatos",
"IP").ToString()
            user = GetSetting("Minerva", "BaseDeDatos",
"Usuario").ToString()
            passwd = GetSetting("Minerva", "BaseDeDatos",
"Contraseña").ToString()
            db = GetSetting("Minerva", "BaseDeDatos", "DB").ToString()
            Dim servidor_sentencia As String = "server=" & server &
";uid=" & user & ";password=" & passwd & ";database=" & db & ";Connect
Timeout=2;"
            Conn = New MySqlConnection(servidor_sentencia)
            Conn.Open()
        Catch ex As Exception
            ' En caso de error mostrar un mensaje y salir
            If Not noSalir Then
                MsgBox("Error al establecer la conexión con el
servidor, puede cambiar los datos de conexión la ventana inicial. El
programa procederá a cerrarse.", MsgBoxStyle.Critical)
                Environment.Exit(0)
            Else
                MsgBox("Error al establecer la conexión con el
servidor, puede cambiar los datos de conexión la ventana inicial.",
MsgBoxStyle.Critical)
            End If
        End Try
    End Sub

    Public Sub Close()
        ' Cierra la conexión
        Conn.Dispose()
    End Sub
End Class
```

9.2.2 PersistenciaAsignaturas.vb

```
Imports MySql.Data.MySqlClient
Imports System.Data

Public Class PersistenciaAsignaturas
    Public Shared Sub Asignar(IdAsignatura As String, NroGrupo As
String, FechaToma As String, GradoAreaProfesor As Integer, Ci As
String)
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "INSERT INTO `Tiene_Ag` VALUES
(@IdAsignatura, @NroGrupo, @FechaToma, @GradoAreaProfesor,
@CiPersona);"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@IdAsignatura",
IdAsignatura)
                .Parameters.AddWithValue("@NroGrupo", NroGrupo)
                .Parameters.AddWithValue("@FechaToma", FechaToma)
                .Parameters.AddWithValue("@GradoAreaProfesor",
GradoAreaProfesor)
                .Parameters.AddWithValue("@CiPersona", Ci)
            End With
            Try
                cmd.ExecuteNonQuery()
                conexion.Close()
            Catch ex As Exception
                conexion.Close()
                Throw ex
            End Try
        End Using
    End Sub

    Public Shared Sub DesAsignar(NroGrupo As String, IdAsignatura As
String, Ci As String)
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "DELETE FROM `Tiene_Ag` WHERE
NroGrupo=@NroGrupo and IdAsignatura=@IdAsignatura and
CiPersona=@CiPersona;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@NroGrupo", NroGrupo)
                .Parameters.AddWithValue("@IdAsignatura",
IdAsignatura)
                .Parameters.AddWithValue("@CiPersona", Ci)
            End With
            Try
                cmd.ExecuteNonQuery()
                conexion.Close()
```

```
            Catch ex As Exception
                conexion.Close()
                Throw ex
            End Try
        End Using
    End Sub

    Public Shared Function GetAsignadasDocente(Ci As String) As Object
        Dim conexion As New Conexion()

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "SELECT IdAsignatura, IdGrupo,
FechaToma, Grado FROM `Tiene_Ag`, `Grupo` WHERE
Tiene_Ag.CiPersona=@CiPersona and Grupo.NroGrupo=Tiene_Ag.NroGrupo;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@CiPersona", Ci)
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function

    Public Shared Function GetAllForGrupo(NroGrupo As String)
        Dim IdOrientacion As String =
PersistenciaGrupos.GetOrientacion(NroGrupo:=NroGrupo)
        Dim Grado As String = PersistenciaGrupos.GetGrado(NroGrupo)

        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "select Tiene_Ta.*, NombreAsignatura
from Tiene_Ta, Asignatura where IdOrientacion=@IdOrientacion and
Grado=@Grado and Tiene_Ta.IdAsignatura=Asignatura.IdAsignatura;"
                .Parameters.AddWithValue("@IdOrientacion",
IdOrientacion)
                .Parameters.AddWithValue("@Grado", Grado)
                .CommandType = CommandType.Text
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function

    Public Shared Function GetProfesor(IdAsignatura As String,
NroGrupo As String) As Object
        Dim NombreProfesor As String = "Sin profesor"
        Dim CiProfesor As String = "-1"
        Dim conexion As New Conexion()
```

```
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "select T.CiPersona,
CONCAT(NombrePersona, ' ', ApellidoPersona) as Profesor from Tiene_Ag
T, Persona P where T.IdAsignatura=@IdAsignatura and
T.NroGrupo=@NroGrupo and P.CiPersona=T.CiPersona;"
                .Parameters.AddWithValue("@IdAsignatura",
IdAsignatura)
                .Parameters.AddWithValue("@NroGrupo", NroGrupo)
                .CommandType = CommandType.Text
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            While reader.Read()
                NombreProfesor = reader("Profesor")
                CiProfesor = reader("CiPersona")
            End While
            reader.Close()
        End Using

        conexion.Close()

        Return {NombreProfesor, CiProfesor}
    End Function
End Class
```

9.2.3 PersistenciaDocentes.vb

```
Imports MySql.Data.MySqlClient
Imports System.Data

Public Class PersistenciaDocentes
    Public Shared Sub Add(Ci As String, GradoProfesor As Integer)
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "INSERT INTO `Profesor` VALUES
(@CiPersona, @GradoProfesor);"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@CiPersona", Ci)
                .Parameters.AddWithValue("@GradoProfesor",
GradoProfesor)
            End With
            Try
                cmd.ExecuteNonQuery()
                conexion.Close()
            Catch ex As Exception
                conexion.Close()
                Throw ex
            End Try
        End Using
    End Sub

    Public Shared Sub Edit(Ci As String, GradoProfesor As Integer)
        Dim conexion As New Conexion()

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandType = CommandType.Text
                .CommandText = "UPDATE `Profesor` SET
GradoProfesor=@GradoProfesor WHERE CiPersona=@CiPersona;"
                .Parameters.AddWithValue("@CiPersona", Ci)
                .Parameters.AddWithValue("@GradoProfesor",
GradoProfesor)
            End With
            cmd.ExecuteNonQuery()
            conexion.Close()
        End Using
    End Sub

    Public Shared Sub Del(Ci As String)
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "DELETE FROM `Profesor` WHERE
CiPersona=@CiPersona;"
```

```
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@CiPersona", Ci)
            End With
            Try
                cmd.ExecuteNonQuery()
                conexion.Close()
            Catch ex As Exception
                conexion.Close()
                Throw ex
            End Try
        End Using
    End Sub

    Public Shared Function GetInfo(Ci As String) As Object
        Dim conexion As New Conexion()

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "SELECT Profesor.*,
Persona.NombrePersona, Persona.ApellidoPersona FROM `Profesor`,
`Persona` where Profesor.CiPersona=@CiPersona and
Profesor.CiPersona=Persona.CiPersona;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@CiPersona", Ci)
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function

    Public Shared Function GetConflictosAsignacion(Ci As String,
IdAsignatura As String, NroGrupo As String) As Object
        Dim Conflictivo As Boolean = False
        Dim IdGrupo As String = ""

        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "select CONCAT(Grado, ' ', IdGrupo) as
Grupo from Genera, Grupo, (select HoraInicio, Dia from Genera where
IdAsignatura=@IdAsignatura and Genera.NroGrupo=@NroGrupo) AEC where
Genera.CiPersona=@CiPersona and Genera.HoraInicio=AEC.HoraInicio and
Genera.Dia=AEC.Dia and Grupo.NroGrupo=@NroGrupo;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@IdAsignatura",
IdAsignatura)
                .Parameters.AddWithValue("@NroGrupo", NroGrupo)
                .Parameters.AddWithValue("@CiPersona", Ci)
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
```

```
            While reader.Read()
                Conflictivo = True
                IdGrupo = reader("Grupo")
            End While
            reader.Close()
            conexion.Close()

            Return {Conflictivo, IdGrupo}
        End Using
    End Function

    Public Shared Function GetConflictoGuardado(HoraInicio As String,
CiPersona As String, Dia As String, Grupo As String, IdAsignatura As
String) As Object
        Dim NombreProfesor As String
        Dim StrGrupo As String
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "select IdAsignatura, Grupo, Dia,
CiPersona, NombreProfesor from Calendario where CiPersona=@CiPersona
and HoraInicio=@horaInicio and Dia=@dia;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@HoraInicio", HoraInicio)
                .Parameters.AddWithValue("@CiPersona", CiPersona)
                .Parameters.AddWithValue("@Dia", Dia)
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            While reader.Read()
                If
reader("IdAsignatura").ToString().Equals(IdAsignatura) And
reader("Grupo").ToString().Equals(Grupo) Then
                    reader.Close()
                    conexion.Close()
                    Return {False}
                End If

                NombreProfesor = reader("NombreProfesor")
                StrGrupo = reader("Grupo")
                reader.Close()
                conexion.Close()
                Return {True, NombreProfesor, StrGrupo}
            End While
            reader.Close()
        End Using

        Return {False}
        conexion.Close()
    End Function
End Class
```

9.2.4 PersistenciaGrupos.vb

```
Imports MySql.Data.MySqlClient
Imports System.Data

Public Class PersistenciaGrupos
    Public Shared Sub Add(IdGrupo As String, Discapacitado As Boolean,
IdSalon As String, IdTurno As Integer, Grado As String, IdOrientacion
As String, CiAdscripto As String)
        Dim conexion As New Conexion()

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandType = CommandType.Text
                .CommandText = "INSERT INTO `Grupo` VALUES (Null,
@IdGrupo, @Discapacitado, @Idsalon, @IdTurno, @Grado, @IDOrientacion,
@CiAdscripto);"
                .Parameters.AddWithValue("@IdGrupo", IdGrupo)
                .Parameters.AddWithValue("@Discapacitado",
Discapacitado)
                .Parameters.AddWithValue("@IdSalon", IdSalon)
                .Parameters.AddWithValue("@IdTurno", IdTurno)
                .Parameters.AddWithValue("@Grado", Grado)
                .Parameters.AddWithValue("@IdOrientacion",
IdOrientacion)
                .Parameters.AddWithValue("@CiAdscripto", CiAdscripto)
            End With

            Try
                cmd.ExecuteNonQuery()
                conexion.Close()
            Catch ex As Exception
                conexion.Close()
                Throw ex
            End Try
        End Using
    End Sub
    Public Shared Sub Edit(IdGrupo As String, Discapacitado As
Boolean, IdSalon As String, IdTurno As Integer, Grado As String,
CiAdscripto As String)
        Dim conexion As New Conexion()

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandType = CommandType.Text
                .CommandText = "UPDATE `Grupo` SET
CiPersona=@CiAdscripto, Discapacitado=@Discapacitado, IdSalon=@IdSalon
where Grado=@Grado and IdGrupo=@IdGrupo"
                .Parameters.AddWithValue("@IdGrupo", IdGrupo)
                .Parameters.AddWithValue("@Discapacitado",
Discapacitado)
```

```
                .Parameters.AddWithValue("@IdSalon", IdSalon)
                .Parameters.AddWithValue("@IdTurno", IdTurno)
                .Parameters.AddWithValue("@Grado", Grado)
                .Parameters.AddWithValue("@CiAdscripto", CiAdscripto)
            End With

            Try
                cmd.ExecuteNonQuery()
                conexion.Close()
            Catch ex As Exception
                conexion.Close()
                Throw ex
            End Try
        End Using
    End Sub

    Public Shared Sub Del(NroGrupo As String)
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "DELETE FROM `Grupo` WHERE
NroGrupo=@NroGrupo;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@NroGrupo", NroGrupo)
            End With
            Try
                cmd.ExecuteNonQuery()
                conexion.Close()
            Catch ex As Exception
                conexion.Close()
                Throw ex
            End Try
        End Using
    End Sub

    Public Shared Function GetInfo(nroGrupo As String) As Object
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "SELECT Grupo.*, Curso.*,
Orientacion.*, CONCAT(Adscriptos.CiPersona, ' - ', Adscripto) as
NombreAdscripto FROM `Grupo`, `Curso`, `Orientacion`, `Adscriptos`
WHERE Grupo.CiPersona=Adscriptos.CiPersona and
Grupo.NroGrupo=@NroGrupo and Orientacion.IdCurso=Curso.IdCurso and
Grupo.IdOrientacion=Orientacion.IdOrientacion;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@NroGrupo", nroGrupo)
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
```

```
    End Function

    Public Shared Function GetNroGrupo(Grupo As String) As String
        Dim NroGrupo As String = Nothing

        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "select * from Grupo where
CONCAT(Grado, ' ', IdGrupo)=@Grupo;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@Grupo", Grupo)
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            While reader.Read()
                NroGrupo = reader("NroGrupo")
            End While
            reader.Close()
            conexion.Close()

            Return NroGrupo
        End Using
    End Function

    Public Shared Function GetExiste(Grupo As String) As Boolean
        Dim conexion As New Conexion()
        Dim existe As Boolean = False

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "select * from Grupo where
CONCAT(Grado, ' ', IdGrupo)=@Grupo"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@Grupo", Grupo)
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            While reader.Read()
                existe = True
            End While
            reader.Close()
        End Using
        conexion.Close()

        Return existe
    End Function

    Public Shared Function GetGrado(NroGrupo As String) As String
        Dim conexion As New Conexion()
        Dim grado As String = ""
```

```
        Dim resultadosPersistencia As Object = GetInfo(NroGrupo)

        Dim reader As MySqlDataReader = resultadosPersistencia(0)
        While reader.Read()
            grado = reader("Grado")
        End While
        reader.Close()

        resultadosPersistencia(1).Close()

        Return grado
    End Function

    Public Shared Function GetOrientacion(Optional IdGrupo As String =
"", Optional Grado As String = "", Optional NroGrupo As String =
Nothing) As String
        Dim IdOrientacion As String = Nothing

        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                If IsNothing(NroGrupo) Then
                    .CommandText = "SELECT IdOrientacion from Grupo
where IdGrupo=@IdGrupo and Grado=@Grado;"
                    .Parameters.AddWithValue("@IdGrupo", IdGrupo)
                    .Parameters.AddWithValue("@Grado", Grado)
                Else
                    .CommandText = "SELECT IdOrientacion from Grupo
where NroGrupo=@NroGrupo;"
                    .Parameters.AddWithValue("@NroGrupo", NroGrupo)
                End If
                .CommandType = CommandType.Text
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            While reader.Read()
                IdOrientacion = reader("IdOrientacion")
            End While
            reader.Close()
            conexion.Close()

            Return IdOrientacion
        End Using
    End Function

    Public Shared Function GetAsignaturaTomada(IdAsignatura As String,
NroGrupo As String) As Boolean
        Dim AsignaturaTomada As Boolean = False

        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
```

```
                .CommandText = "SELECT * from `Tiene_Ag` WHERE
IdAsignatura=@IdAsignatura and Tiene_Ag.NroGrupo=@NroGrupo;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@IdAsignatura",
IdAsignatura)
                .Parameters.AddWithValue("@NroGrupo", NroGrupo)
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            While reader.Read()
                AsignaturaTomada = True
            End While
            reader.Close()
            conexion.Close()

            Return AsignaturaTomada
        End Using
    End Function

    Public Shared Function GetDetalles(NroGrupo As String) As Object
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "select A.Adscripto, D.*, T.IdTurno,
CONCAT(G.Grado, ' ', G.IdGrupo) as Grupo from DatosGrupos D, Turno T,
Adscriptos A, Grupo G where A.CiPersona=D.CiPersona and
T.NombreTurno=D.NombreTurno and D.Grado=G.Grado and
D.IdGrupo=G.IdGrupo and G.NroGrupo=@NroGrupo;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@NroGrupo", NroGrupo)
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function

    Public Shared Function GetTurno(StringGrupo As String) As String
        Dim IdTurno As String = Nothing
        Dim NroGrupo As String = GetNroGrupo(StringGrupo)

        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "SELECT IdTurno from Grupo where
NroGrupo=@NroGrupo;"
                .Parameters.AddWithValue("@NroGrupo", NroGrupo)
                .CommandType = CommandType.Text
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            While reader.Read()
```

```
                IdTurno = reader("IdTurno")
            End While
            reader.Close()
            conexion.Close()

            Return IdTurno
        End Using
    End Function
End Class
```

9.2.5 PersistenciaHorarios.vb

```
Imports MySql.Data.MySqlClient
Imports System.Data

Public Class PersistenciaHorarios
    Public Shared Sub Add(IdAsignatura As String, NroGrupo As String,
HoraInicio As String, HoraFin As String, Dia As String, IdTurno As
String, CiPersona As String)
        Dim conexion As New Conexion()

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandType = CommandType.Text
                .CommandText = "INSERT INTO `Genera` VALUES
(@IdAsignatura, @NroGrupo, @HoraInicio, @HoraFin, @Dia, @IdTurno,
@CiPersona);"
                .Parameters.AddWithValue("@IdAsignatura",
IdAsignatura)
                .Parameters.AddWithValue("@NroGrupo", NroGrupo)
                .Parameters.AddWithValue("@HoraInicio", HoraInicio)
                .Parameters.AddWithValue("@HoraFin", HoraFin)
                .Parameters.AddWithValue("@Dia", Dia)
                .Parameters.AddWithValue("@IdTurno", IdTurno)
                .Parameters.AddWithValue("@CiPersona", CiPersona)
            End With
            Try
                cmd.ExecuteNonQuery()
                conexion.Close()
            Catch ex As Exception
                conexion.Close()
                Throw ex
            End Try
        End Using
    End Sub

    Public Shared Sub ForceDel(HoraInicio As String, HoraFin As
String, Dia As String, NroGrupo As String, IdTurno As String)
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "DELETE FROM `Genera` WHERE
HoraInicio=@HoraInicio and HoraFin=@HoraFin and Dia=@Dia and
NroGrupo=@NroGrupo and IdTurno=@IdTurno"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@HoraInicio", HoraInicio)
                .Parameters.AddWithValue("@HoraFin", HoraFin)
                .Parameters.AddWithValue("@Dia", Dia)
                .Parameters.AddWithValue("@NroGrupo", NroGrupo)
                .Parameters.AddWithValue("@IdTurno", IdTurno)
            End With
```

```
            Try
                cmd.ExecuteNonQuery()
                conexion.Close()
            Catch ex As Exception
                conexion.Close()
                Throw ex
            End Try
        End Using
    End Sub

    Public Shared Sub Del(IdAsignatura As String, NroGrupo As String,
CiPersona As String)
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "UPDATE `Genera` SET `CiPersona`='-1'
WHERE `IdAsignatura`=@IdAsignatura and `NroGrupo`=@NroGrupo and
`CiPersona`=@CiPersona;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@NroGrupo", NroGrupo)
                .Parameters.AddWithValue("@IdAsignatura",
IdAsignatura)
                .Parameters.AddWithValue("@CiPersona", CiPersona)
            End With
            Try
                cmd.ExecuteNonQuery()
                conexion.Close()
            Catch ex As Exception
                conexion.Close()
                Throw ex
            End Try
        End Using
    End Sub

    Public Shared Sub Edit(Ci As String, IdAsignatura As String,
NroGrupo As String)
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "UPDATE `Genera` SET
`CiPersona`=@CiPersona WHERE `IdAsignatura`=@IdAsignatura and
`NroGrupo`=@NroGrupo;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@CiPersona", Ci)
                .Parameters.AddWithValue("@IdAsignatura",
IdAsignatura)
                .Parameters.AddWithValue("@NroGrupo", NroGrupo)
            End With
            Try
                cmd.ExecuteNonQuery()
                conexion.Close()
            Catch ex As Exception
```

```
                conexion.Close()
                Throw ex
            End Try
        End Using
    End Sub

    Public Shared Function GetForGrupo(NroGrupo As String) As Object
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "SELECT * FROM Genera WHERE
`NroGrupo`=@NroGrupo;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@NroGrupo", NroGrupo)
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function

    Public Shared Sub DelGrupoEntero(NroGrupo As String)
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "DELETE FROM Genera WHERE
`NroGrupo`=@NroGrupo;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@NroGrupo", NroGrupo)
            End With
            Try
                cmd.ExecuteNonQuery()
                conexion.Close()
            Catch ex As Exception
                conexion.Close()
                Throw ex
            End Try
        End Using
    End Sub

    Public Shared Function GetCalendarioForGrupo(ByVal StringGrupo As
String) As Object
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "select * from Calendario where
Grupo=@StringGrupo;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@StringGrupo", StringGrupo)
            End With
```

```
            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function

    Public Shared Function GetCalendarioDiarioForGrupo(Dia As String,
StringGrupo As String) As Object
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "select *, DATE_FORMAT(HoraInicio,
'%H:%i') as HoraOrden from Calendario where Dia=@Dia and
Grupo=@StringGrupo order by HoraOrden;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@StringGrupo", StringGrupo)
                .Parameters.AddWithValue("@Dia", Dia)
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function

    Public Shared Function GetHorariosTurno(IdTurno As String) As
Object
        Dim conexion As New Conexion()

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "select DISTINCT HoraInicio, HoraFin
from Asignacion where IdTurno=@IdTurno;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@IdTurno", IdTurno)
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using

    End Function

    Public Shared Function GetForProfesor(CiProfesor As String,
IdTurno As Integer) As Object
        Dim conexion As New Conexion()

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "select Calendario.* from Calendario,
Grupo where CONCAT(Grupo.Grado, ' ', Grupo.IdGrupo)=Calendario.Grupo
and Calendario.CiPersona=@CiPersona and Grupo.IdTurno=@IdTurno;"
                .CommandType = CommandType.Text
```

```
                .Parameters.AddWithValue("@CiPersona", CiProfesor)
                .Parameters.AddWithValue("@IdTurno", IdTurno)
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function
End Class
```

9.2.6 PersistenciaPersonas.vb

```
Imports MySql.Data.MySqlClient
Imports System.Data

Public Class PersistenciaPersonas

    Public Shared Sub Add(Ci As String, Optional Nombre As String =
Nothing, Optional Apellido As String = Nothing)
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "INSERT INTO `Persona` VALUES
(@CiPersona, @Nombre, @Apellido);"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@CiPersona", Ci)
                .Parameters.AddWithValue("@Nombre", Nombre)
                .Parameters.AddWithValue("@Apellido", Apellido)
            End With
            Try
                cmd.ExecuteNonQuery()
                conexion.Close()
            Catch ex As Exception
                conexion.Close()
                Throw ex
            End Try
        End Using
    End Sub

    Public Shared Sub Edit(Ci As String, Nombre As String, Apellido As
String)
        Dim conexion As New Conexion()

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandType = CommandType.Text
                .CommandText = "UPDATE `Persona` SET
NombrePersona=@NombrePersona, ApellidoPersona=@ApellidoPersona WHERE
CiPersona=@CiPersona;"

                .Parameters.AddWithValue("@CiPersona", Ci)
                .Parameters.AddWithValue("@NombrePersona", Nombre)
                .Parameters.AddWithValue("@ApellidoPersona", Apellido)
            End With
            cmd.ExecuteNonQuery()
            conexion.Close()
        End Using
    End Sub

    Public Shared Sub Del(Ci As String)
        Dim conexion As New Conexion()
```

```
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "DELETE FROM `Persona` WHERE
CiPersona=@CiPersona"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@CiPersona", Ci)
            End With
            Try
                cmd.ExecuteNonQuery()
                conexion.Close()
            Catch ex As Exception
                conexion.Close()
                Throw ex
            End Try
        End Using
    End Sub
End Class
```

9.2.7 PersistenciaSalones.vb

```
Imports MySql.Data.MySqlClient
Imports System.Data

Public Class PersistenciaSalones

    Public Shared Sub Add(IdSalon As String, ComentariosSalon As
String, PlantaSalon As String)
        Dim conexion As New Conexion()

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandType = CommandType.Text
                .CommandText = "INSERT INTO `Salon` VALUES (@IdSalon,
@ComentariosSalon, @PlantaSalon);"

                .Parameters.AddWithValue("@IdSalon", IdSalon)
                .Parameters.AddWithValue("@ComentariosSalon",
ComentariosSalon)
                .Parameters.AddWithValue("@PlantaSalon", PlantaSalon)
            End With

            Try
                cmd.ExecuteNonQuery()
                conexion.Close()
            Catch ex As Exception
                conexion.Close()
                Throw ex
            End Try
        End Using
    End Sub

    Public Shared Sub Edit(IdSalon As String, ComentariosSalon As
String, PlantaSalon As String)
        Dim conexion As New Conexion()

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandType = CommandType.Text
                .CommandText = "UPDATE `Salon` SET
ComentariosSalon=@ComentariosSalon, PlantaSalon=@PlantaSalon WHERE
IDSalon=@IDSalon;"

                .Parameters.AddWithValue("@IdSalon", IdSalon)
                .Parameters.AddWithValue("@ComentariosSalon",
ComentariosSalon)
                .Parameters.AddWithValue("@PlantaSalon", PlantaSalon)
            End With

            Try
```

```
                cmd.ExecuteNonQuery()
                conexion.Close()
            Catch ex As Exception
                conexion.Close()
                Throw ex
            End Try
        End Using
    End Sub

    Public Shared Function GetInfo(IdSalon As String) As Object
        Dim conexion As New Conexion()

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "SELECT * FROM `Salon` where
IdSalon=@IdSalon;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@IdSalon", IdSalon)
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function

    Public Shared Function GetAsignado(IdSalon As String, Turno As
Integer) As Object
        Dim conexion As New Conexion()

        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "SELECT CONCAT(Grupo.Grado, ' ',
Grupo.IdGrupo) as Grupo FROM `Salon`, `Grupo` WHERE
Salon.IdSalon=Grupo.IdSalon and Grupo.IdTurno=@IdTurno and
Salon.IdSalon=@IdSalon;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@IdSalon", IdSalon)
                .Parameters.AddWithValue("@IdTurno", Turno)
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function

    Public Shared Sub Del(IdSalon As String)
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "DELETE FROM `Salon` WHERE
IdSalon=@IdSalon;"
```

```
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@IdSalon", IdSalon)
            End With
            Try
                cmd.ExecuteNonQuery()
                conexion.Close()
            Catch ex As Exception
                conexion.Close()
                Throw ex
            End Try
        End Using
    End Sub

    Public Shared Function GetOcupadoBy(IdSalon As String, Turno As
Integer) As Object
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "select * from Grupo where
IdSalon=@IdSalon and IdTurno=@IdTurno and not IdSalon='-1';"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@IdSalon", IdSalon)
                .Parameters.AddWithValue("@IdTurno", Turno)
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function
End Class
```

9.2.8 PersistenciaUsuario.vb

```
Imports MySql.Data.MySqlClient
Imports System.Data

Public Class PersistenciaUsuarios

    Public Shared Sub Add(Ci As String, TipoUsuario As String,
ContraseniaUsuario As String, AprobacionUsuario As Boolean)
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "INSERT INTO `Usuario` VALUES
(@CiPersona, @TipoUsuario, @ContraseniaUsuario, @AprobacionUsuario);"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@CiPersona", Ci)
                .Parameters.AddWithValue("@TipoUsuario", TipoUsuario)
                .Parameters.AddWithValue("@ContraseniaUsuario",
ContraseniaUsuario)
                .Parameters.AddWithValue("@AprobacionUsuario",
AprobacionUsuario)
            End With
            Try
                cmd.ExecuteNonQuery()
                conexion.Close()
            Catch ex As Exception
                conexion.Close()
                Throw ex
            End Try
        End Using
    End Sub

    Public Shared Sub Del(Ci As String)
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "DELETE FROM `Usuario` WHERE
CiPersona=@CiPersona"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@CiPersona", Ci)
            End With
            Try
                cmd.ExecuteNonQuery()
                conexion.Close()
            Catch ex As Exception
                conexion.Close()
                Throw ex
            End Try
        End Using
    End Sub
```

```
    Public Shared Sub Edit(Ci As String, TipoUsuario As String,
ContraseniaUsuario As String, AprobacionUsuario As Boolean)
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "UPDATE `Usuario` SET
TipoUsuario=@TipoUsuario, ContraseniaUsuario=@ContraseniaUsuario,
AprobacionUsuario=@AprobacionUsuario WHERE CiPersona=@CiPersona;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@CiPersona", Ci)
                .Parameters.AddWithValue("@TipoUsuario", TipoUsuario)
                .Parameters.AddWithValue("@ContraseniaUsuario",
ContraseniaUsuario)
                .Parameters.AddWithValue("@AprobacionUsuario",
AprobacionUsuario)
            End With
            Try
                cmd.ExecuteNonQuery()
                conexion.Close()
            Catch ex As Exception
                conexion.Close()
                Throw ex
            End Try
        End Using
    End Sub

    Public Shared Function GetInfo(Ci As String) As Object
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "SELECT Usuario.*, NombrePersona,
ApellidoPersona from Usuario, Persona where
Usuario.CiPersona=Persona.CiPersona and Usuario.CiPersona=@Ci;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@Ci", Ci)
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function

    Public Shared Function GetNombre(Ci As String) As Object
        Dim Nombre As String = ""
        Dim Apellido As String = ""

        Dim resultadosPersistencia As Object = GetInfo(Ci)
        Dim reader As MySqlDataReader = resultadosPersistencia(0)
        While reader.Read()
            Try
                Nombre = reader("NombrePersona")
                Apellido = reader("ApellidoPersona")
```

```
            Catch ex As Exception
            End Try
        End While

        Return {Nombre, Apellido}
    End Function

    Public Shared Function Login(Ci As String, Contraseña As String)
As Object
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "SELECT * FROM `Usuario` WHERE
CiPersona=@Ci AND ContraseniaUsuario=@Contrasenia;"
                .CommandType = CommandType.Text
                .Parameters.AddWithValue("@Ci", Ci)
                .Parameters.AddWithValue("@Contrasenia", Contraseña)
            End With

            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            Return {reader, conexion}
        End Using
    End Function

    Public Shared Function CantAdministradores() As Integer
        Dim cantidadAdministradores As Integer
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
                .CommandText = "select COUNT(*) as 'Cantidad' from
Usuario WHERE Usuario.TipoUsuario='Administrador' and
Usuario.AprobacionUsuario is True;"
                .CommandType = CommandType.Text
            End With
            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            While reader.Read()
                cantidadAdministradores =
Integer.Parse(reader("Cantidad"))
            End While
            reader.Close()
        End Using
        conexion.Close()
        Return cantidadAdministradores
    End Function

    Public Shared Function CantSinAprobar() As Integer
        Dim cantidadAprobar As Integer = 0
        Dim conexion As New Conexion()
        Using cmd As New MySqlCommand()
            With cmd
                .Connection = conexion.Conn
```

```
                .CommandText = "select COUNT(*) as 'Cantidad' from
Usuario WHERE Usuario.AprobacionUsuario=False and not
Usuario.CiPersona='-1';"
                .CommandType = CommandType.Text
            End With
            Dim reader As MySqlDataReader = cmd.ExecuteReader()
            While reader.Read()
                cantidadAprobar = Integer.Parse(reader("Cantidad"))
            End While
            reader.Close()
        End Using
        conexion.Close()

        Return cantidadAprobar
    End Function
End Class
```

9.3 Capa de presentación

9.3.1 frmAcerva.b

```
Public Class frmAcerca

    Private Sub btnVolver_Click(sender As Object, e As EventArgs)
Handles btnVolver.Click
        Me.Dispose()
    End Sub
End Class
```

9.3.2 frmAdminDocentes.vb

```
Public Class frmAdminDocentes
    ' Clase principal para la administracion de docentes

    Friend totalDocentes As Integer = 0

    Dim frmMain As frmMain
    Dim prevSelect As String
    Dim prevGrupoSelect As String
    Dim docentePreview As Object = New Button()

    Public Sub New(frmMain As frmMain)
        InitializeComponent()
        Me.frmMain = frmMain
    End Sub

    Private Sub frmAdminDocentes_Load(sender As Object, e As
EventArgs) Handles MyBase.Load
        lstAsignaturas.FullRowSelect = True
        Docente.CargarDocentes(Me)
        Docente.CargarGrupos(Me)
        txtCI.Focus()
    End Sub

    Public Sub AgregarWidgetDocente(ciDocente As String, txtDocente As
String)
        Dim pnlTemporal As New Panel
        Dim btnDocente As New Button
        Dim btnEditar, btnEliminar As New Button

        pnlTemporal.Size = pnlDocentePlantilla.Size
        btnDocente.Size = btnDocentePlantilla.Size
        btnDocente.FlatStyle = btnDocentePlantilla.FlatStyle
        btnDocente.ForeColor = btnDocentePlantilla.ForeColor
        btnDocente.Text = txtDocente
        btnDocente.BackColor = btnDocentePlantilla.BackColor
        btnDocente.Location = btnDocentePlantilla.Location
        btnDocente.Cursor = btnDocentePlantilla.Cursor
        btnDocente.Font = btnDocentePlantilla.Font
        btnDocente.TabStop = False

        btnDocente.Tag = ciDocente
        AddHandler btnDocente.Click, AddressOf InterfazVerDocente

        btnEditar.Size = btnEditarPlantilla.Size
        btnEditar.FlatStyle = btnEditarPlantilla.FlatStyle
        btnEditar.ForeColor = btnEditarPlantilla.ForeColor
        btnEditar.Text = btnEditarPlantilla.Text
        btnEditar.BackColor = btnEditarPlantilla.BackColor
        btnEditar.Location = btnEditarPlantilla.Location
        btnEditar.Cursor = btnEditarPlantilla.Cursor
        btnEditar.TabStop = False
```

```
        btnEditar.Tag = ciDocente
        AddHandler btnEditar.MouseUp, AddressOf
AbrirMenuEdicionDocente

        btnEliminar.Size = btnEliminarPlantilla.Size
        btnEliminar.FlatStyle = btnEliminarPlantilla.FlatStyle
        btnEliminar.ForeColor = btnEliminarPlantilla.ForeColor
        btnEliminar.Text = btnEliminarPlantilla.Text
        btnEliminar.BackColor = btnEliminarPlantilla.BackColor
        btnEliminar.Location = btnEliminarPlantilla.Location
        btnEliminar.Cursor = btnEliminarPlantilla.Cursor
        btnEliminar.TabStop = False

        btnEliminar.Tag = {ciDocente,
txtDocente.Replace(ControlChars.NewLine, " ")}
        AddHandler btnEliminar.Click, AddressOf EliminarDocente

        AddHandler pnlTemporal.MouseWheel, AddressOf fixScroll
        AddHandler pnlTemporal.MouseEnter, AddressOf fixScroll
        pnlTemporal.Controls.Add(btnDocente)
        pnlTemporal.Controls.Add(btnEditar)
        pnlTemporal.Controls.Add(btnEliminar)

        pnlDocentes.Controls.Add(pnlTemporal)

        totalDocentes += 1
        lblCantidadDocentes.Text = "(" + totalDocentes.ToString() +
")"
    End Sub

    Private Sub fixScroll(sender As Object, e As Object)
        pnlDocentes.Focus()
    End Sub

    Private Sub AbrirMenuEdicionDocente(sender As System.Object, e As
MouseEventArgs) Handles btnEditarPlantilla.MouseUp
        DatosDelDocenteToolStripMenuItem.Tag = {sender.Parent,
sender.Tag}
        AsignaturasDelDocenteToolStripMenuItem.Tag = {sender.Parent,
sender.Tag}
        mnuEdicionDocente.Show(sender, New Point(e.X, e.Y))
    End Sub

    Private Sub CheckDatosDocente(Optional sender As Object = Nothing,
Optional e As EventArgs = Nothing) Handles btnAgregarDocente.Click
        If String.IsNullOrWhiteSpace(txtCI.Text) Then
            MessageBox.Show("Debe ingresar una CI.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
            Return
        End If

        If String.IsNullOrWhiteSpace(txtNombre.Text) Then
```

```
            MessageBox.Show("Debe ingresar un nombre.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
            Return
        End If

        If String.IsNullOrWhiteSpace(txtApellido.Text) Then
            MessageBox.Show("Debe ingresar un apellido.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
            Return
        End If

        Docente.ActualizarDB(Me)
        frmMain.recargarGrupo()
    End Sub

    Private Sub InterfazEditarDocente(sender As System.Object)
        docentePreview.Enabled = True
        docentePreview = sender.Tag(0)
        docentePreview.Enabled = False
        lblNuevoDocente.Text = "Editar docente"
        btnCancelarEdicion.Visible = True
        btnAgregarDocente.Visible = True
        btnAgregarAsignatura.Visible = False
        btnLimpiarAsignatura.Visible = False
        btnEliminarAsignatura.Visible = False
        btnNuevoDocente.Visible = False
        btnAgregarDocente.Text = "Guardar cambios"

        HabilitarControles(True)
        Docente.CargarInfo(sender.Tag(1), Me)
        txtCI.Enabled = False
        HabilitarControlesAsignaturas(False)
        Docente.CargarAsignaturas(sender.Tag(1), Me)

        btnHorariosDocente.Visible = False
    End Sub

    Private Sub HabilitarControles(habilitado As Boolean)
        txtCI.Enabled = habilitado
        txtCI.Text = ""
        txtNombre.Enabled = habilitado
        txtNombre.Text = ""
        txtApellido.Enabled = habilitado
        txtApellido.Text = ""
        numGrado.Enabled = habilitado
        numGrado.Value = 1
    End Sub

    Private Sub HabilitarControlesAsignaturas(habilitadas As Boolean)
        lstAsignaturas.Enabled = habilitadas
        lstAsignaturas.Items.Clear()
        btnLimpiarAsignatura.Enabled = habilitadas
        btnEliminarAsignatura.Enabled = habilitadas
        cmbArea.Enabled = False
```

```
        cmbArea.SelectedIndex = -1
        cmbAsignatura.Enabled = False
        cmbAsignatura.SelectedIndex = -1
        cmbGrupo.Enabled = habilitadas
        cmbGrupo.SelectedIndex = -1
        numGradoArea.Enabled = habilitadas
        numGradoArea.Value = 1
        btnAgregarAsignatura.Enabled = habilitadas
    End Sub

    Private Sub InterfazVerDocente(sender As System.Object, e As
System.EventArgs)
        docentePreview.Enabled = True
        docentePreview = sender
        docentePreview.Enabled = False
        InterfazPrevisualizarDocente(sender.Tag)

        lstAsignaturas.Enabled = True
        Docente.CargarAsignaturas(sender.Tag, Me)
        btnHorariosDocente.Visible = True
    End Sub

    Public Sub InterfazPrevisualizarDocente(ci As String)
        btnNuevoDocente.Visible = True
        btnAgregarDocente.Visible = False
        btnAgregarAsignatura.Visible = False
        btnCancelarEdicion.Visible = False
        btnLimpiarAsignatura.Visible = False
        btnEliminarAsignatura.Visible = False

        lblNuevoDocente.Text = "Previsualizar docente"
        HabilitarControlesAsignaturas(False)

        HabilitarControles(False)
        Docente.CargarInfo(ci, Me)

        docentePreview.Enabled = True
        docentePreview = Nothing
        For Each pnl As Panel In pnlDocentes.Controls
            For Each x As Button In pnl.Controls
                If x.Tag.Equals(ci) Then
                    If IsNothing(docentePreview) Then
                        docentePreview = x
                    End If
                End If
            Next
        Next
        docentePreview.Enabled = False
        btnHorariosDocente.Visible = True
    End Sub

    Public Sub InterfazNuevoDocente(Optional sender As Object =
Nothing, Optional e As EventArgs = Nothing) Handles
btnNuevoDocente.Click, btnCancelarEdicion.Click
```

```
        HabilitarControles(True)
        lblNuevoDocente.Text = "Nuevo docente"
        btnNuevoDocente.Visible = False
        btnAgregarAsignatura.Visible = False
        btnAgregarDocente.Visible = True
        btnLimpiarAsignatura.Visible = False
        btnEliminarAsignatura.Visible = False
        btnAgregarDocente.Text = "Agregar docente " &
ControlChars.NewLine & "y editar asignaturas"
        btnCancelarEdicion.Visible = False
        docentePreview.Enabled = True
        docentePreview = New Button()
        LimpiarDatosAsignatura()
        HabilitarControlesAsignaturas(False)
        btnHorariosDocente.Visible = False
    End Sub

    Private Sub EditarDatosDelDocente_Click(sender As Object, e As
EventArgs) Handles DatosDelDocenteToolStripMenuItem.Click
        InterfazEditarDocente(sender)
    End Sub

    Public Sub LimpiarDatosAsignatura(Optional sender As Object =
Nothing, Optional e As EventArgs = Nothing) Handles
btnLimpiarAsignatura.Click
        cmbArea.SelectedIndex = -1
        cmbGrupo.SelectedIndex = -1
        numGradoArea.Value = 1
    End Sub

    Private Sub GrupoCambiado(sender As Object, e As EventArgs)
Handles cmbGrupo.SelectedIndexChanged
        If cmbGrupo.Text.Equals(prevGrupoSelect) Then
            Return
        End If
        cmbArea.SelectedIndex = -1

        prevGrupoSelect = cmbGrupo.Text
        cmbArea.Enabled = False
        If Not String.IsNullOrWhiteSpace(cmbGrupo.Text) Then
            cmbArea.Enabled = True
        Else
            Return
        End If
        Docente.CargarAreas(Me)
    End Sub

    Private Sub AreaCambiada(sender As Object, e As EventArgs) Handles
cmbArea.SelectedIndexChanged
        If cmbArea.Text.Equals(prevSelect) Then
            Return
        End If
        cmbAsignatura.SelectedIndex = -1
```

```
        prevSelect = cmbArea.Text
        cmbAsignatura.Enabled = False
        If Not String.IsNullOrWhiteSpace(cmbArea.Text) Then
            cmbAsignatura.Enabled = True
        Else
            Return
        End If
        Docente.CargarAsignaturasGrupo(Me)
    End Sub

    Private Sub CheckDatosAsignatura(Optional sender As Object =
Nothing, Optional e As EventArgs = Nothing) Handles
btnAgregarAsignatura.Click
        If String.IsNullOrWhiteSpace(cmbArea.Text) Then
            MessageBox.Show("Debe seleccionar un área.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
            Return
        End If

        If String.IsNullOrWhiteSpace(cmbAsignatura.Text) Then
            MessageBox.Show("Debe seleccionar una asignatura.",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
            Return
        End If

        If String.IsNullOrWhiteSpace(cmbGrupo.Text) Then
            MessageBox.Show("Debe seleccionar un grupo.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
            Return
        End If

        Docente.ActualizarDB_Asignatura(Me)
        frmMain.RecargarGrupo()
    End Sub

    Private Sub EditarAsignaturasDelDocente_Click(sender As Object, e
As EventArgs) Handles AsignaturasDelDocenteToolStripMenuItem.Click
        ' Deshabilita la edición de datos del docente.
        InterfazEditarAsignaturasDocente(sender.Tag(1))
        docentePreview.Enabled = True
        docentePreview = sender.Tag(0)
        docentePreview.Enabled = False
    End Sub

    Public Sub InterfazEditarAsignaturasDocente(CI As String)
        InterfazPrevisualizarDocente(CI)
        HabilitarControlesAsignaturas(True)
        btnAgregarAsignatura.Visible = True
        btnLimpiarAsignatura.Visible = True
        btnEliminarAsignatura.Visible = False
        lstAsignaturas.Enabled = True
        Docente.CargarAsignaturas(CI, Me)
        lblNuevoDocente.Text = "Editar asignaturas del docente"
        docentePreview.Enabled = True
```

```
        docentePreview = docentePreview.Parent
        docentePreview.Enabled = False
        btnHorariosDocente.Visible = True
    End Sub

    Private Sub EliminarAsignatura(sender As Object, e As EventArgs)
Handles btnEliminarAsignatura.Click
        Dim result As Integer = MessageBox.Show("¿Está seguro de que
desea eliminar la asignatura seleccionada?", "Eliminar asignatura",
MessageBoxButtons.YesNo, MessageBoxIcon.Question)
        If result = DialogResult.No Then
            Return
        End If

        Docente.EliminarAsignatura(Me)
        frmMain.RecargarGrupo()
    End Sub

    Private Sub EliminarDocente(sender As Object, e As EventArgs)
        Dim result As Integer = MessageBox.Show("¿Está seguro de que
desea eliminar el docente '" + sender.Tag(1) + "'?", "Eliminar
docente", MessageBoxButtons.YesNo, MessageBoxIcon.Question)
        If result = DialogResult.No Then
            Return
        End If
        Docente.EliminarDocente(sender.Tag(0), sender.Tag(1), Me)
        frmMain.RecargarGrupo()
    End Sub

    Private Sub AsignaturaSeleccionada(sender As Object, e As
EventArgs) Handles lstAsignaturas.SelectedIndexChanged
        If btnLimpiarAsignatura.Visible = False Then
            Return
        End If
        btnEliminarAsignatura.Visible = False
        If lstAsignaturas.SelectedItems.Count > 0 Then
            btnEliminarAsignatura.Visible = True
        End If
    End Sub

    Private Sub CheckNumeros(t As Object, e As KeyEventArgs) Handles
txtCI.KeyDown
        If e.KeyCode.Equals(Keys.Delete) Or
e.KeyCode.Equals(Keys.Back) Or e.KeyCode.Equals(Keys.Left) Or
e.KeyCode.Equals(Keys.Right) Or e.KeyCode.Equals(Keys.Tab) Then
            e.Handled = False
            Return
        End If
        If Not Char.IsDigit(Chr(e.KeyValue)) Then
            e.SuppressKeyPress = True
        End If

    End Sub
```

```
    Private Sub Animacion_1_L(sender As Object, e As EventArgs)
Handles btnLimpiarAsignatura.MouseLeave
        btnLimpiarAsignatura.BackgroundImage =
My.Resources.agregar_normal()
        pnlAyudabtnAgregarAsignatura.Visible = False
    End Sub

    Private Sub Animacion_1_E(sender As Object, e As EventArgs)
Handles btnLimpiarAsignatura.MouseEnter
        ' al entrar a el botón btnAgregarAsignatura cambiar la imagen
        btnLimpiarAsignatura.BackgroundImage =
My.Resources.agregar_hover()
        pnlAyudabtnAgregarAsignatura.Visible = True
    End Sub

    Private Sub Animacion_2_L(sender As Object, e As EventArgs)
Handles btnEliminarAsignatura.MouseLeave
        btnEliminarAsignatura.BackgroundImage =
My.Resources.borrar_normal()
        pnlAyudabtnEliminarAsignatura.Visible = False
    End Sub

    Private Sub Animacion_2_E(sender As Object, e As EventArgs)
Handles btnEliminarAsignatura.MouseEnter
        btnEliminarAsignatura.BackgroundImage =
My.Resources.borrar_hover()
        pnlAyudabtnEliminarAsignatura.Visible = True
    End Sub

    Private Sub btnHorariosDocente_click(sender As Object, e As
EventArgs) Handles btnHorariosDocente.Click
        Docente.CrearGrillaHorariosDocente(Me)
    End Sub
End Class
```

9.3.3 frmAdminGrupos.vb

```
Public Class frmAdminGrupos
    ' Clase principal para la administración de grupos

    Friend CiUsuario As String
    Friend frmAdministrar As frmAdministrar
    Friend frmMain As frmMain
    Friend PrimerGrupo As String
    Friend TipoUsuario As String
    Friend TotalGrupos As Integer = 0

    Dim PrevSelect As String
    Dim PrevOrientacionSelect As String

    Dim NroGrupo As String
    Dim GrupoPreview As Control = New Button()
    Dim Editando As Boolean = False
    Dim Previsualizando As Boolean = False

    Public Sub New(frmMain As frmMain, tipousuario As String,
ciUsuario As String, frmAdministrar As frmAdministrar)
        InitializeComponent()
        Me.frmMain = frmMain
        Me.TipoUsuario = tipousuario
        Me.CiUsuario = ciUsuario
        Me.frmAdministrar = frmAdministrar
    End Sub

    Private Sub frmAdminGrupos_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
        ' Al cargar la ventana, cargarGrupos y rellenar los combos con
los datos.

        HabilitarControles(True)

        Grupo.CargarTurnos(Me)
        Grupo.CargarCursos(Me)
        Grupo.CargarAdscriptos(Me)
        Grupo.CargarSalones(Me)

        cmbAdscripto.SelectedIndex = 0

        cmbOrientacion.Enabled = False
        cboGrado.Enabled = False
        cmbCurso.Focus()
        CargarGrupos()
    End Sub

    Public Sub AgregarWidgetGrupo(NroGrupo As String, idTexto As
String, nombreTurno As String, ciAdscripto As String)
        Dim pnlTemporal As New Panel
        Dim btnGrupo As New Button
```

```
Dim btnEliminar As New Button
Dim btnEditar As New Button()

pnlTemporal.Size = pnlGrupoPlantilla.Size
btnGrupo.Size = btnGrupoPlantilla.Size
btnGrupo.FlatStyle = btnGrupoPlantilla.FlatStyle
btnGrupo.ForeColor = btnGrupoPlantilla.ForeColor
btnGrupo.Text = idTexto
btnGrupo.Margin = btnGrupoPlantilla.Margin
btnGrupo.TextAlign = btnGrupoPlantilla.TextAlign
btnGrupo.BackColor = btnGrupoPlantilla.BackColor
btnGrupo.Location = btnGrupoPlantilla.Location
btnGrupo.Cursor = btnGrupoPlantilla.Cursor
btnGrupo.Font = btnGrupoPlantilla.Font
btnGrupo.TabStop = False

btnGrupo.Tag = {NroGrupo, "principal"}
AddHandler btnGrupo.Click, AddressOf ClickVerGrupo

btnEditar.Size = btnEditarPlantilla.Size
btnEditar.FlatStyle = btnEditarPlantilla.FlatStyle
btnEditar.ForeColor = btnEditarPlantilla.ForeColor
btnEditar.Text = btnEditarPlantilla.Text
btnEditar.BackColor = btnEditarPlantilla.BackColor
btnEditar.Location = btnEditarPlantilla.Location
btnEditar.Cursor = btnEditarPlantilla.Cursor
btnEditar.TabStop = False
btnEditar.Tag = {NroGrupo, "editar"}
AddHandler btnEditar.Click, AddressOf InterfazEditarGrupo

btnEliminar.Size = btnEliminarPlantilla.Size
btnEliminar.FlatStyle = btnEliminarPlantilla.FlatStyle
btnEliminar.ForeColor = btnEliminarPlantilla.ForeColor
btnEliminar.Text = btnEliminarPlantilla.Text
btnEliminar.BackColor = btnEliminarPlantilla.BackColor
btnEliminar.Location = btnEliminarPlantilla.Location
btnEliminar.Cursor = btnEliminarPlantilla.Cursor
btnEliminar.TabStop = False

btnEliminar.Tag = {NroGrupo, idTexto, nombreTurno}
AddHandler btnEliminar.Click, AddressOf EliminarGrupo

If Me.TipoUsuario.Equals("Adscripto") Then
    btnEliminar.Visible = False
    btnEditar.Text = "Editar" & vbCrLf & "salón"
    btnEditar.Height = btnGrupo.Height
End If
AddHandler pnlTemporal.MouseWheel, AddressOf fixScroll
AddHandler pnlTemporal.MouseEnter, AddressOf fixScroll
pnlTemporal.Controls.Add(btnEditar)
pnlTemporal.Controls.Add(btnEliminar)
pnlTemporal.Controls.Add(btnGrupo)
pnlGrupos.Controls.Add(pnlTemporal)
```

```
        TotalGrupos += 1
        lblCantidadGrupos.Text = "(" + TotalGrupos.ToString() + ")"
    End Sub

    Private Sub fixScroll(sender As Object, e As Object)
        pnlGrupos.Focus()
    End Sub

    Private Sub HabilitarControles(habilitado As Boolean)
        ' Habilita o deshabilita los controles
        txtIdGrupo.Enabled = habilitado
        txtIdGrupo.Text = ""

        cmbTurno.Enabled = habilitado
        cmbTurno.SelectedIndex = -1

        PrevSelect = ""
        cmbCurso.Enabled = habilitado
        cmbCurso.SelectedIndex = -1
        cmbOrientacion.Enabled = habilitado
        cmbOrientacion.SelectedIndex = -1
        cboGrado.Enabled = habilitado
        cboGrado.SelectedIndex = -1
        chkDiscapacitado.Enabled = True
        chkDiscapacitado.Checked = False

        cmbSalon.Enabled = habilitado
    End Sub

    Public Sub InterfazNuevoGrupo(Optional sender As Object = Nothing,
Optional e As EventArgs = Nothing) Handles btnNuevoGrupo.Click
        ' Prepara la interfaz para agregar un nuevo grupo
        If Me.TipoUsuario.Equals("Adscripto") Then
            InterfazPrevisualizarGrupo(NroGrupo)
            Return
        End If
        HabilitarControles(True)
        lblNuevoGrupo.Text = "Nuevo grupo"
        btnNuevoGrupo.Visible = False
        btnAgregar.Visible = True
        btnAgregar.Text = "Agregar grupo"
        cmbOrientacion.Enabled = False
        cboGrado.Enabled = False
        Editando = False
        Previsualizando = False
        chkDiscapacitado.TabStop = True
        cmbAdscripto.Enabled = True

        Grupo.CargarTurnos(Me)
        Grupo.CargarCursos(Me)
        Grupo.CargarAdscriptos(Me)
        Grupo.CargarSalones(Me)

        GrupoPreview.Enabled = True
```

```
        GrupoPreview = New Button()
        chkDistribuir.Checked = True
        chkDistribuir.Visible = True
    End Sub

    Private Sub CheckDatosCorrectos(Optional sender As Object =
Nothing, Optional e As EventArgs = Nothing) Handles btnAgregar.Click
        ' Comprueba los datos y en caso de que no falte ninguno, llama
a actualizarDB()
        If Me.TipoUsuario.Equals("Adscripto") And Not Editando Then
            MessageBox.Show("Oops!" & vbCrLf & "Solo los
administradores y funcionarios pueden hacer eso...", "Acceso
denegado", MessageBoxButtons.OK, MessageBoxIcon.Warning)
            Return
        End If

        If String.IsNullOrWhiteSpace(txtIdGrupo.Text) Then
            MessageBox.Show("Debe ingresar un ID de grupo.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
            Return
        End If

        If String.IsNullOrWhiteSpace(cmbCurso.Text) Then
            MessageBox.Show("Debe seleccionar un curso.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
            Return
        End If

        If String.IsNullOrWhiteSpace(cmbOrientacion.Text) Then
            MessageBox.Show("Debe seleccionar una orientación.",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
            Return
        End If

        If String.IsNullOrWhiteSpace(cmbTurno.Text) Then
            MessageBox.Show("Debe seleccionar un turno", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
            Return
        End If

        If String.IsNullOrWhiteSpace(cboGrado.Text) Then
            MessageBox.Show("Debe seleccionar un grado", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
            Return
        End If

        Grupo.ActualizarDB(Me)
    End Sub

    Private Sub ClickVerGrupo(sender As System.Object, e As
System.EventArgs)
        InterfazPrevisualizarGrupo(sender.Tag(0))

        chkDiscapacitado.TabStop = False
```

```
    End Sub

    Private Sub InterfazEditarGrupo(sender As System.Object, e As
System.EventArgs)
        Dim nroGrupo As String = sender.Tag(0)
        InterfazPrevisualizarGrupo(sender.Tag(0))

        GrupoPreview.Enabled = True
        GrupoPreview = sender.Parent
        GrupoPreview.Enabled = False

        btnNuevoGrupo.Text = "Nuevo grupo / cancelar"
        If Me.TipoUsuario.Equals("Adscripto") Then
            btnNuevoGrupo.Text = "Cancelar"
        End If
        btnNuevoGrupo.Visible = True
        btnAgregar.Text = "Guardar cambios"
        btnAgregar.Visible = True
        lblNuevoGrupo.Text = "Editar grupo"
        cmbAdscripto.Enabled = True
        Previsualizando = False
        Editando = True
        cmbSalon.Enabled = True
        If Me.TipoUsuario.Equals("Adscripto") Then
            cmbAdscripto.Enabled = False
        End If
        chkDistribuir.Checked = False
        chkDistribuir.Visible = False
    End Sub

    Private Sub InterfazPrevisualizarGrupo(nroGrupo As String)
        ' Prepara la interfaz para previsualizarUnGrupo
        Me.NroGrupo = nroGrupo
        btnNuevoGrupo.Visible = True
        btnAgregar.Visible = False
        btnNuevoGrupo.Text = "Nuevo grupo"
        lblNuevoGrupo.Text = "Previsualizar grupo"

        GrupoPreview.Enabled = True
        GrupoPreview = Nothing
        For Each pnl As Panel In pnlGrupos.Controls
            For Each x As Button In pnl.Controls
                If x.Tag(0).Equals(nroGrupo) And
x.Tag(1).Equals("principal") Then
                    If IsNothing(GrupoPreview) Then
                        GrupoPreview = x
                    End If
                End If
            Next
        Next

        If Me.TipoUsuario.Equals("Adscripto") Then
            btnNuevoGrupo.Visible = False
        End If
```

```
        GrupoPreview.Enabled = False

        chkDistribuir.Checked = False
        chkDistribuir.Visible = False

        HabilitarControles(False)
        Grupo.CargarGrupo(nroGrupo, Me)
        Previsualizando = True
    End Sub

    Private Sub CursoCambiado(sender As Object, e As EventArgs)
Handles cmbCurso.SelectedIndexChanged
        ' Al cambiar el id de curso, cargarOrientaciones
        If cmbCurso.Text.Equals(PrevSelect) Then
            Return
        End If
        cmbOrientacion.SelectedIndex = -1
        Grupo.CargarOrientaciones(Me)
        cmbOrientacion.Enabled = False
        If Not String.IsNullOrWhiteSpace(cmbCurso.Text) Then
            cmbOrientacion.Enabled = True
        End If
        PrevSelect = cmbCurso.Text
    End Sub

    Private Sub IgnorarEspacio(sender As System.Object, e As
System.Windows.Forms.KeyPressEventArgs) Handles txtIdGrupo.KeyPress
        ' Al apretar la tecla espacio, ignorarla
        If e.KeyChar = " " Then
            e.KeyChar = Nothing
        End If
    End Sub

    Private Sub CargarGrupos()
        Grupo.CargarGrupos(Me)

        If Me.TotalGrupos = 0 And Me.TipoUsuario.Equals("Adscripto")
Then
            lblNoGrupoAsignado.Visible = True
        ElseIf Me.TipoUsuario.Equals("Adscripto") Then
            InterfazPrevisualizarGrupo(PrimerGrupo)
        End If
    End Sub

    Private Sub EliminarGrupo(sender As System.Object, e As
System.EventArgs)
        ' Le pregunta al usuario si quiere eliminar el grupo, de ser
correcto, lo elimina
        Dim grupo_ As String
        grupo_ = sender.Tag(1)
        Dim result As Integer = MessageBox.Show("¿Está seguro de que
desea eliminar el grupo '" + grupo_ + "'?", "Eliminar grupo",
MessageBoxButtons.YesNo, MessageBoxIcon.Question)
        If result = DialogResult.No Then
```

```
            Return
        End If

        Grupo.EliminarGrupo(sender.Tag(0), sender.Tag(1), Me)
        Me.frmMain.RecargarGrupo()
    End Sub

    Private Sub IngnorarClick_ChkDiscapacitado(sender As Object, e As
EventArgs) Handles chkDiscapacitado.Click
        If Previsualizando Or Me.TipoUsuario.Equals("Adscripto") Then
            chkDiscapacitado.Checked = Not chkDiscapacitado.Checked
        End If
    End Sub

    Private Sub OrientacionCambiada(sender As Object, e As EventArgs)
Handles cmbOrientacion.SelectedIndexChanged
        If cmbOrientacion.Text.Equals(PrevOrientacionSelect) Then
            Return
        End If

        cboGrado.SelectedIndex = -1
        Grupo.CargarGrados(Me)
        cboGrado.Enabled = False
        If Not String.IsNullOrWhiteSpace(cmbOrientacion.Text) Then
            cboGrado.Enabled = True
        End If
        PrevOrientacionSelect = cmbOrientacion.Text
    End Sub

End Class
```

9.3.4 frmAdminHorarios.vb

```
Public Class frmAdminHorarios
    ' Clase principal para la administración de horarios
    Dim prevHover As Control = New Control()
    Dim prevSelect As String = Nothing

    Friend _IdTurno As String = Nothing
    Friend horarioPrimera As String = "13:00"
    Friend finPrimera As String = "13:45"
    Friend horarioSegunda As String = "13:50"
    Friend finSegunda As String = "14:35"
    Friend horarioTercera As String = "14:40"
    Friend finTercera As String = "15:25"
    Friend horarioCuarta As String = "15:30"
    Friend finCuarta As String = "16:15"
    Friend horarioQuinta As String = "16:20"
    Friend finQuinta As String = "17:05"
    Friend horarioSexta As String = "17:10"
    Friend finSexta As String = "17:55"
    Friend horarioExtra As String = "18:00"
    Friend finExtra As String = "18:45"
    Friend frmMain As frmMain
    Friend frmAdministrar As frmAdministrar
    Dim tablas As Object = Nothing

    Public Sub New(frmMain As frmMain, frmAdministrar As
frmAdministrar)
        Me.frmMain = frmMain
        Me.frmAdministrar = frmAdministrar
        ' Llamada necesaria para el diseñador.
        InitializeComponent()

        ' Agregue cualquier inicialización después de la llamada a
InitializeComponent().
    End Sub

    Public Sub Materia_MouseDown(sender As Object, e As
MouseEventArgs)
        Dim btn As Control = TryCast(sender, Control)
        btn.DoDragDrop(btn, DragDropEffects.Move)
        prevHover.BackColor = Color.FromArgb(35, 35, 35)
        If (e.Button = Windows.Forms.MouseButtons.Right) Then
            If Not (sender.Parent Is pnlMaterias Or
sender.Text.Equals("Sin asignar")) Then
                sender.Parent.BackColor = Color.FromArgb(35, 35, 35)
                sender.Parent.Controls.Remove(sender)
                pnlMaterias.Controls.Add(sender)
            ElseIf (sender.Text.Equals("Sin asignar") And Not
(sender.Parent Is pnlMaterias)) Then
                sender.Parent.Controls.Remove(sender)
            End If
```

```
        End If
    End Sub

    Private Sub pnl_DragDrop(sender As Object, e As DragEventArgs)
        Dim c As Control =
TryCast(e.Data.GetData(e.Data.GetFormats()(0)), Control)

        prevHover.BackColor = Color.FromArgb(35, 35, 35)
        If TypeOf (c) Is TableLayoutPanel Then
            c = c.Controls(0)
        End If
        If c IsNot Nothing Then
            c.Location = sender.PointToClient(New Point(e.X, e.Y))
            sender.Controls.Add(c)
        End If
    End Sub

    Public Sub fixScroll(sender As Object, e As Object)
        pnlMaterias.Focus()
    End Sub

    Private Sub pnl_DragOver(sender As Object, e As DragEventArgs)
        prevHover.BackColor = Color.FromArgb(35, 35, 35)
        sender.BackColor = Color.FromArgb(50, 50, 50)
        If sender.Controls.Count > 0 And Not (sender Is pnlMaterias)
Then
            e.Effect = DragDropEffects.None
        Else
            e.Effect = DragDropEffects.Move
        End If
        prevHover = sender
    End Sub


    Private Sub frmAdminHorarios_Load(sender As Object, e As
EventArgs) Handles MyBase.Load
        For Each c As Control In pnlMaterias.Controls
            AddHandler c.MouseDown, AddressOf Materia_MouseDown
        Next

        tablas = {
            tableLunes1, tableLunes2, tableLunes3, tableLunes4,
tableLunes5, tableLunes6, tableLunes7, _
            tableMartes1, tableMartes2, tableMartes3, tableMartes4,
tableMartes5, tableMartes6, tableMartes7, _
            tableMiercoles1, tableMiercoles2, tableMiercoles3,
tableMiercoles4, tableMiercoles5, tableMiercoles6, tableMiercoles7, _
            tableJueves1, tableJueves2, tableJueves3, tableJueves4,
tableJueves5, tableJueves6, tableJueves7, _
            tableViernes1, tableViernes2, tableViernes3,
tableViernes4, tableViernes5, tableViernes6, tableViernes7, _
            tableSabado1, tableSabado2, tableSabado3, tableSabado4,
tableSabado5, tableSabado6, tableSabado7, pnlMaterias
        }
```

```
        For Each tabla As Control In tablas
            tabla.AllowDrop = True
            AddHandler tabla.DragOver, AddressOf pnl_DragOver
            AddHandler tabla.DragDrop, AddressOf pnl_DragDrop
            For Each c As Control In tabla.Controls
                AddHandler c.MouseDown, AddressOf Materia_MouseDown
            Next
        Next

        FuncionesHorarios.CargarGrupos(Me)
        cmbGrupo.Focus()
    End Sub

    Private Sub cmbGrupo_SelectedIndexChanged(sender As Object, e As
EventArgs) Handles cmbGrupo.SelectedIndexChanged
        If cmbGrupo.Text.Equals(prevSelect) Then
            Return
        End If

        prevSelect = cmbGrupo.Text

        cargarGrupo()
    End Sub

    Private Sub btnGuardado_Click(sender As Object, e As EventArgs)
Handles btnGuardado.Click
        dialogoEspere.BringToFront()
        FuncionesHorarios.GuardarHorarios(Me)
        dialogoEspere.SendToBack()
    End Sub

    Public Sub LimpiarTablas()
        For Each tabla As Control In tablas
            tabla.Controls.Clear()
        Next
    End Sub
    Private Sub cargarGrupo()
        LimpiarTablas()

        lblSeleccioneGrupo.Visible = False
        lblTapaMaterias.Visible = False
        pnlMaterias.Enabled = True
        btnDeshacer.Visible = True

        btnGuardado.TabStop = True
        btnLimpiar.TabStop = True

        If cmbGrupo.Text.Equals("...") Then
            lblSeleccioneGrupo.Visible = True
            lblTapaMaterias.Visible = True
            pnlMaterias.Enabled = False
            btnDeshacer.Visible = False
            btnGuardado.TabStop = False
```

```
            btnLimpiar.TabStop = False
            Return
        End If

        dialogoEspere.BringToFront()

        FuncionesHorarios.CargarHorariosTurno(Me)
        FuncionesHorarios.CargarAsignaturas(Me)
        dialogoEspere.SendToBack()
    End Sub

    Private Sub btnLimpiar_Click(sender As Object, e As EventArgs)
Handles btnLimpiar.Click
        For Each tabla As Control In tablas
            If tabla Is pnlMaterias Then
                Return
            End If
            For Each btn As Control In tabla.Controls()
                btn.Parent = Nothing
                pnlMaterias.Controls.Add(btn)
            Next
        Next
    End Sub

    Public Sub actHorarios()
        lblHora1.Text = horarioPrimera.Substring(0,
horarioPrimera.Length - 3)
        lblHora2.Text = horarioSegunda.Substring(0,
horarioSegunda.Length - 3)
        lblHora3.Text = horarioTercera.Substring(0,
horarioTercera.Length - 3)
        lblHora4.Text = horarioCuarta.Substring(0,
horarioCuarta.Length - 3)
        lblHora5.Text = horarioQuinta.Substring(0,
horarioQuinta.Length - 3)
        lblHora6.Text = horarioSexta.Substring(0, horarioSexta.Length
- 3)
        lblHora7.Text = horarioExtra.Substring(0, horarioExtra.Length
- 3)
    End Sub

    Private Sub btnDeshacer_Click(sender As Object, e As EventArgs)
Handles btnDeshacer.Click
        cargarGrupo()
    End Sub

    Private Sub btnDeshacer_Leave(sender As Object, e As EventArgs)
Handles btnDeshacer.MouseLeave
        ' al dejar el botón btnDeshacer cambiar la imagen
        btnDeshacer.BackgroundImage = My.Resources.cancelar_normal()
        pnlAyudaBtnDeshacer.Visible = False
    End Sub
```

```
    Private Sub btnDeshacer_Enter(sender As Object, e As EventArgs)
Handles btnDeshacer.MouseEnter
        ' al entrar a el botón btnDeshacer cambiar la imagen
        btnDeshacer.BackgroundImage = My.Resources.cancelar_hover()
        pnlAyudaBtnDeshacer.Visible = True
    End Sub

End Class
```

9.3.5 frmAdministrar.vb

```
Imports MySql.Data.MySqlClient

Public Class frmAdministrar
    ' Clase principal de administración de datos

    Dim btnActual As Button = New Button()
    Dim pnlTrabajo As New UserControl
    Dim tipoUsuario As String
    Dim frmMain As frmMain
    Dim abrirUsuarios As Boolean

    Public Sub New(tipoUsuario As String, frmMain As frmMain, Optional
abrirUsuarios As Boolean = False)
        InitializeComponent()
        Me.tipoUsuario = tipoUsuario
        Me.frmMain = frmMain
        Me.abrirUsuarios = abrirUsuarios
    End Sub

    Public Sub habilitarBotones(habilitar As Boolean)
        btnSalones.Enabled = habilitar
        btnGrupos.Enabled = habilitar
        btnDocentes.Enabled = habilitar
        btnHorarios.Enabled = habilitar
        btnUsuarios.Enabled = habilitar
    End Sub

    Private Sub frmAdministrar_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
        If tipoUsuario.Equals("Administrador") Then
            btnSalones.Visible = True
            btnGrupos.Visible = True
            btnDocentes.Visible = True
            btnHorarios.Visible = True
            btnUsuarios.Visible = True
            btnBackup.Visible = True
            btnLimpiar.Visible = False
        ElseIf tipoUsuario.Equals("Funcionario") Then
            btnSalones.Visible = True
            btnGrupos.Visible = True
            btnDocentes.Visible = True
            btnHorarios.Visible = True
            btnUsuarios.Visible = False
            btnBackup.Visible = False
            btnLimpiar.Visible = False
        ElseIf tipoUsuario.Equals("Adscripto") Then
            btnSalones.Visible = True
            btnGrupos.Visible = True
            btnDocentes.Visible = False
            btnHorarios.Visible = False
            btnUsuarios.Visible = False
```

```
            btnBackup.Visible = False
            btnLimpiar.Visible = False
        End If
        ' Clickear el btnSalones por defecto al iniciar
        btnSalones.PerformClick()
        If Me.abrirUsuarios Then
            btnUsuarios.PerformClick()
        End If
    End Sub

    Private Sub botones_Click(sender As Object, e As EventArgs)
Handles btnSalones.Click, btnGrupos.Click, btnDocentes.Click,
btnHorarios.Click, btnUsuarios.Click
        ' Cambia la pestaña seleccionada y le cambia el fondo
        pnlTrabajo.Focus()
        If btnActual Is sender Then
            Return
        End If

        btnActual.Location = New Point(btnActual.Location.X, 4)
        btnActual.BackColor = Color.FromArgb(45, 45, 45)
        btnActual.FlatAppearance.BorderColor = Color.Gray
        btnActual.FlatAppearance.MouseDownBackColor =
Color.FromArgb(45, 45, 45)
        btnActual.FlatAppearance.MouseOverBackColor =
Color.FromArgb(45, 45, 45)
        btnActual.Cursor = Cursors.Hand

        btnActual = sender
        btnActual.Location = New Point(btnActual.Location.X, 0)
        btnActual.BackColor = Color.FromArgb(35, 35, 35)
        btnActual.FlatAppearance.BorderColor = Color.PaleGreen
        btnActual.FlatAppearance.MouseDownBackColor =
Color.FromArgb(35, 35, 35)
        btnActual.FlatAppearance.MouseOverBackColor =
Color.FromArgb(35, 35, 35)
        btnActual.Cursor = Cursors.Default

        pnlBorde.BringToFront()
        btnActual.BringToFront()
        pnlTrabajo.BringToFront()
    End Sub
    Private Sub acomodarDiseño(Optional original As Boolean = True)
        If original Then
            pnlBorde.Size = New Point(1007, 2)
            pnlBorde1.Location = New Point(1005, 41)
            pnlBorde1.Size = New Point(3, 504)
            pnlBorde2.Location = New Point(0, 40)
            pnlBorde2.Size = New Point(2, 500)
            pnlBorde3.Location = New Point(0, 535)
            pnlBorde3.Size = New Point(1007, 2)

            pnlBorde.BringToFront()
            pnlBorde1.BringToFront()
```

```
            pnlBorde2.BringToFront()
            pnlBorde3.BringToFront()
        Else
            pnlBorde.Size = New Point(1400, 2)

            pnlBorde1.Location = New Point(1182, 41)
            pnlBorde1.Size = New Point(3, 604)

            pnlBorde2.Location = New Point(0, 40)
            pnlBorde2.Size = New Point(2, 600)

            pnlBorde3.Location = New Point(0, 560)
            pnlBorde3.Size = New Point(1200, 2)

            pnlBorde.BringToFront()
            pnlBorde1.BringToFront()
            pnlBorde2.BringToFront()
            pnlBorde3.BringToFront()
        End If
    End Sub

    Private Sub Centrar()
        '' Note: call this from frm's Load event!
        Dim r As Rectangle
        r = Me.frmMain.RectangleToScreen(frmMain.ClientRectangle)

        Dim x = r.Left + (r.Width - Me.Width) \ 2
        Dim y = r.Top + (r.Height - Me.Height) \ 2
        Me.Location = New Point(x, y)
    End Sub

    ' Al clickear X boton mostrar X panel.

    Private Sub btnSalones_Click(sender As Object, e As EventArgs)
Handles btnSalones.Click
        If btnActual Is sender Then
            If TypeOf (pnlTrabajo) Is frmAdminSalones Then
                Return
            End If
        End If

        Me.Text = "Minerva · Administración de salones"
        Me.Controls.Remove(pnlTrabajo)
        pnlTrabajo = New frmAdminSalones(Me.tipoUsuario)
        Me.Controls.Add(pnlTrabajo)
        Me.Size = New Point(1024, 575)
        pnlTrabajo.Location = New Point(2, 42)
        pnlTrabajo.BringToFront()
        sender.BringToFront()
        acomodarDiseño()
        Centrar()
    End Sub
```

```
    Private Sub btnGrupos_Click(sender As Object, e As EventArgs)
Handles btnGrupos.Click
        If btnActual Is sender Then
            If TypeOf (pnlTrabajo) Is frmAdminGrupos Then
                Return
            End If
        End If

        Me.Text = "Minerva · Administración de grupos"
        Me.Controls.Remove(pnlTrabajo)
        acomodarDiseño()
        pnlTrabajo = New frmAdminGrupos(Me.frmMain, Me.tipoUsuario,
Me.frmMain.NombreUsuario, Me)
        Me.Size = New Point(1024, 575)
        Me.Controls.Add(pnlTrabajo)
        pnlTrabajo.Location = New Point(2, 42)
        pnlTrabajo.BringToFront()
        sender.BringToFront()
        Centrar()
    End Sub

    Private Sub btnDocentes_Click(sender As Object, e As EventArgs)
Handles btnDocentes.Click
        If btnActual Is sender Then
            If TypeOf (pnlTrabajo) Is frmAdminDocentes Then
                Return
            End If
        End If

        Me.Text = "Minerva · Administración de docentes"
        Me.Controls.Remove(pnlTrabajo)
        acomodarDiseño()
        pnlTrabajo = New frmAdminDocentes(Me.frmMain)

        Me.Size = New Point(1024, 575)
        Me.Controls.Add(pnlTrabajo)
        pnlTrabajo.Location = New Point(2, 42)
        pnlTrabajo.BringToFront()
        sender.BringToFront()
        Centrar()
    End Sub

    Private Sub btnHorarios_Click(sender As Object, e As EventArgs)
Handles btnHorarios.Click
        If btnActual Is sender Then
            If TypeOf (pnlTrabajo) Is frmAdminHorarios Then
                Return
            End If
        End If

        Me.Text = "Minerva · Administración de horarios"
        Me.Controls.Remove(pnlTrabajo)
        acomodarDiseño(False)
        pnlTrabajo = New frmAdminHorarios(Me.frmMain, Me)
```

```
        Me.Size = New Point(1200, 600)
        Me.Controls.Add(pnlTrabajo)
        pnlTrabajo.Location = New Point(2, 42)
        pnlTrabajo.BringToFront()
        sender.BringToFront()
        Centrar()
    End Sub

    Private Sub btnUsuarios_Click(sender As Object, e As EventArgs)
Handles btnUsuarios.Click
        If btnActual Is sender Then
            If TypeOf (pnlTrabajo) Is frmAdminUsuarios Then
                Return
            End If
        End If

        Me.Text = "Minerva · Administración de usuarios"
        Me.Controls.Remove(pnlTrabajo)
        acomodarDiseño()
        pnlTrabajo = New frmAdminUsuarios(Me.frmMain.NombreUsuario,
Me.frmMain)
        Me.Size = New Point(1024, 575)
        Me.Controls.Add(pnlTrabajo)
        pnlTrabajo.Location = New Point(2, 42)
        pnlTrabajo.BringToFront()
        sender.BringToFront()
        Centrar()
    End Sub

    Private Sub frmAdministrar_FormClosing(sender As Object, e As
FormClosingEventArgs) Handles MyBase.FormClosing
        Me.Hide()
        FuncionesMinerva.CargarNombre(Me.frmMain)
        frmMain.RecargarGrupo()
    End Sub

    Private Sub btnBackup_Click(sender As Object, e As EventArgs)
Handles btnBackup.Click
        Dim path As String = Nothing
        If sfdBackupSQL.ShowDialog = Windows.Forms.DialogResult.OK
Then
            path = sfdBackupSQL.FileName
            Try
                Dim conexion As New Conexion()
                Using cmd As New MySqlCommand()
                    Using backup As New MySqlBackup(cmd)
                        cmd.Connection = conexion.Conn
                        backup.ExportToFile(path)
                        conexion.Close()
                        MessageBox.Show("Respaldo guardado.", "Base de
datos", MessageBoxButtons.OK, MessageBoxIcon.Information)
                    End Using
                End Using
            Catch ex As Exception
```

```
                MessageBox.Show("Error al respaldar.", "Base de
datos", MessageBoxButtons.OK, MessageBoxIcon.Information)
            End Try
        End If
    End Sub
    ' btnBackup
    Private Sub Animacion_1_E(sender As Object, e As EventArgs)
Handles btnBackup.MouseEnter
        pnlAyudabtnBackup.Visible = True
        pnlAyudabtnBackup.BringToFront()
        sender.BackgroundImage = My.Resources.guardar_sql_hover()
    End Sub

    Private Sub Animacion_1_L(sender As Object, e As EventArgs)
Handles btnBackup.MouseLeave
        pnlAyudabtnBackup.Visible = False
        sender.BackgroundImage = My.Resources.guardar_sql_normal()
    End Sub

    Private Sub btnLimpiar_Click(sender As Object, e As EventArgs)
Handles btnLimpiar.Click
        Dim frmLimpiarDB As New frmLimpiarDB()
        frmLimpiarDB.ShowDialog(Me)
    End Sub
End Class
```

9.3.6 frmAdminSalones.vb

```
Public Class frmAdminSalones
    Friend PrimerSalon As String
    Friend TotalSalones As Integer = 0

    Dim TipoUsuario As String
    Dim Editando As Boolean = False
    Dim SalonPreview As Object = New Button()

    Public Sub New(tipousuario As String)
        InitializeComponent()

        Me.TipoUsuario = tipousuario
    End Sub

    Private Sub frmAdminSalones_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
        HabilitarControles(True)
        txtIdSalon.Focus()
        CargarSalones()
    End Sub

    Public Sub AgregarWidgetSalon(idSalon As String)
        Dim pnlTemporal As New Panel
        Dim btnSalon As New Button
        Dim btnEditar, btnEliminar As New Button

        pnlTemporal.Size = pnlSalonPlantilla.Size
        btnSalon.Size = btnSalonPlantilla.Size
        btnSalon.FlatStyle = btnSalonPlantilla.FlatStyle
        btnSalon.ForeColor = btnSalonPlantilla.ForeColor
        btnSalon.Text = idSalon
        btnSalon.BackColor = btnSalonPlantilla.BackColor
        btnSalon.Location = btnSalonPlantilla.Location
        btnSalon.Cursor = btnSalonPlantilla.Cursor
        btnSalon.Font = btnSalonPlantilla.Font
        btnSalon.TabStop = False

        btnSalon.Tag = idSalon
        AddHandler btnSalon.Click, AddressOf ClickVerSalon

        btnEditar.Size = btnEditarPlantilla.Size
        btnEditar.FlatStyle = btnEditarPlantilla.FlatStyle
        btnEditar.ForeColor = btnEditarPlantilla.ForeColor
        btnEditar.Text = btnEditarPlantilla.Text
        btnEditar.BackColor = btnEditarPlantilla.BackColor
        btnEditar.Location = btnEditarPlantilla.Location
        btnEditar.Cursor = btnEditarPlantilla.Cursor
        btnEditar.TabStop = False

        btnEditar.Tag = idSalon
        AddHandler btnEditar.Click, AddressOf InterfazEditarSalon
```

```
        btnEliminar.Size = btnEliminarPlantilla.Size
        btnEliminar.FlatStyle = btnEliminarPlantilla.FlatStyle
        btnEliminar.ForeColor = btnEliminarPlantilla.ForeColor
        btnEliminar.Text = btnEliminarPlantilla.Text
        btnEliminar.BackColor = btnEliminarPlantilla.BackColor
        btnEliminar.Location = btnEliminarPlantilla.Location
        btnEliminar.Cursor = btnEliminarPlantilla.Cursor
        btnEliminar.TabStop = False

        btnEliminar.Tag = idSalon
        If Me.TipoUsuario.Equals("Adscripto") Then
            btnEliminar.Visible = False
            btnEditar.Visible = False
            btnSalon.Width = btnSalon.Width + btnEliminar.Width
        End If
        AddHandler btnEliminar.Click, AddressOf EliminarSalon

        pnlTemporal.Controls.Add(btnEditar)
        pnlTemporal.Controls.Add(btnEliminar)
        pnlTemporal.Controls.Add(btnSalon)

        AddHandler pnlTemporal.MouseEnter, AddressOf fixScroll
        AddHandler pnlTemporal.MouseWheel, AddressOf fixScroll
        pnlSalones.Controls.Add(pnlTemporal)

        TotalSalones += 1
        lblCantidadSalones.Text = "(" + TotalSalones.ToString() + ")"
    End Sub

    Private Sub fixScroll(sender As Object, e As Object)
        pnlSalones.Focus()
    End Sub

    Private Sub InterfazEditarSalon(sender As System.Object, e As
System.EventArgs)
        SalonPreview.Enabled = True
        SalonPreview = sender.Parent
        SalonPreview.Enabled = False

        lblNuevoSalon.Text = "Editar salón"
        btnAgregar.Text = "Guardar cambios"

        btnCancelarEdicion.Visible = True
        btnAgregar.Visible = True
        btnNuevoSalon.Visible = False

        HabilitarControles(True)

        txtIdSalon.Enabled = False
        cmbPlanta.Enabled = False
        Editando = True
        pnlAsignado.Visible = False
```

```
        Salon.CargarSalon(sender.Tag, Me)
    End Sub

    Public Sub InterfazPrevisualizarSalon(IdSalon As String)
        btnNuevoSalon.Visible = True
        btnAgregar.Visible = False
        btnCancelarEdicion.Visible = False

        SalonPreview.Enabled = True
        SalonPreview = Nothing
        For Each pnl As Panel In pnlSalones.Controls
            For Each x As Button In pnl.Controls
                If x.Text.Equals(IdSalon) Then
                    If IsNothing(SalonPreview) Then
                        SalonPreview = x
                    End If
                End If
            Next
        Next

        SalonPreview.Enabled = False

        lblNuevoSalon.Text = "Previsualizar salón"
        pnlAsignado.Visible = True

        HabilitarControles(False)
        Salon.CargarSalon(IdSalon, Me)
        If Me.TipoUsuario.Equals("Adscripto") Then
            btnNuevoSalon.Visible = False
        End If
    End Sub

    Private Sub HabilitarControles(habilitado As Boolean)
        txtIdSalon.Enabled = habilitado
        txtIdSalon.Text = ""
        cmbPlanta.Enabled = habilitado
        cmbPlanta.SelectedIndex = -1
        lblSalonMatutino.Text = "Sin asignar"
        lblSalonNocturno.Text = "Sin asignar"
        lblSalonVespertino.Text = "Sin asignar"
        txtComentarios.Enabled = habilitado
        txtComentarios.Text = ""
    End Sub

    Public Sub IntefazNuevoSalon(Optional sender As Object = Nothing,
Optional e As EventArgs = Nothing) Handles btnNuevoSalon.Click,
btnCancelarEdicion.Click
        HabilitarControles(True)

        btnNuevoSalon.Visible = False
        btnAgregar.Visible = True
        btnCancelarEdicion.Visible = False
        pnlAsignado.Visible = False
```

```
        lblNuevoSalon.Text = "Nuevo salón"
        btnAgregar.Text = "Agregar salón"

        SalonPreview.Enabled = True
        SalonPreview = New Button()
        Editando = False
    End Sub

    Private Sub CheckNumeros(t As Object, e As KeyEventArgs) Handles
txtIdSalon.KeyDown
        If e.KeyCode.Equals(Keys.Delete) Or
e.KeyCode.Equals(Keys.Back) Or e.KeyCode.Equals(Keys.Left) Or
e.KeyCode.Equals(Keys.Right) Or e.KeyCode.Equals(Keys.Tab) Then
            e.Handled = False
            Return
        End If
        If Not Char.IsDigit(Chr(e.KeyValue)) Then
            e.SuppressKeyPress = True
        End If
    End Sub

    Private Sub ClickVerSalon(sender As System.Object, e As
System.EventArgs)
        InterfazPrevisualizarSalon(sender.Tag)
    End Sub

    ' Comunicación con lógica y persistencia
    Private Sub CheckDatosCorrectos(Optional sender As Object =
Nothing, Optional e As EventArgs = Nothing) Handles btnAgregar.Click
        If Me.TipoUsuario.Equals("Adscripto") And Not Editando Then
            MessageBox.Show("Oops!" & vbCrLf & "Solo los
administradores y funcionarios pueden hacer eso...", "Acceso
denegado", MessageBoxButtons.OK, MessageBoxIcon.Stop)
            Return
        End If

        If String.IsNullOrWhiteSpace(txtIdSalon.Text) Then
            MessageBox.Show("Debe ingresar un ID de salón.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Stop)
            Return
        End If

        If String.IsNullOrWhiteSpace(cmbPlanta.Text) Then
            MessageBox.Show("Debe ingresar la planta del salón.",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Stop)
            Return
        End If

        Salon.ActualizarDB(Me)
    End Sub

    Public Sub CargarSalones()
        Salon.CargarSalones(Me)
```

```
        If Me.TotalSalones = 0 And Me.TipoUsuario.Equals("Adscripto")
Then
            lblAunNoHaySalones.Visible = True
        ElseIf Me.TipoUsuario.Equals("Adscripto") Then
            InterfazPrevisualizarSalon(PrimerSalon)
        End If
    End Sub

    Private Sub EliminarSalon(sender As System.Object, e As
System.EventArgs)
        Dim result As Integer = MessageBox.Show("¿Está seguro de que
desea eliminar el salón '" + sender.Tag + "'?", "Eliminar salón",
MessageBoxButtons.YesNo, MessageBoxIcon.Question)
        If result = DialogResult.No Then
            Return
        End If
        Salon.EliminarSalon(sender.Tag, Me)
    End Sub
End Class
```

9.3.7 frmAdminUsuarios.vb

```
Public Class frmAdminUsuarios
    Friend totalUsuarios As Integer = 0
    Friend tipoSeleccionado As String = "Funcionario"

    Dim previsualizando As Boolean = False
    Dim miUsuario As String = ""
    Dim frmMain As frmMain

    Public Sub New(usuario As String, frmMain As frmMain)
        ' Llamada necesaria para el diseñador.
        InitializeComponent()
        Me.miUsuario = usuario
        Me.frmMain = frmMain
    End Sub

    Private Sub frmAdminUsuarios_Load(sender As Object, e As
EventArgs) Handles MyBase.Load
        ' Al iniciar el programa, cargar los usuarios, y reiniciar la
interfaz
        Usuario.CargarUsuarios(Me)
        InterfazNuevoUsuario()
        txtID.Focus()
    End Sub

    Public Sub AgregarWidgetUsuario(IDUsuario As String, Tipo As
String, usuarioAprobado As Boolean)
        ' Basicamente copio la plantilla a un nuevo panel
        Dim pnlTemporal As New Panel
        Dim btnUsuario As New Button
        Dim btnEliminar, btnEditar As New Button

        pnlTemporal.Size = pnlUsuarioPlantilla.Size
        btnUsuario.Size = btnUsuarioPlantilla.Size
        btnUsuario.FlatStyle = btnUsuarioPlantilla.FlatStyle
        btnUsuario.ForeColor = btnUsuarioPlantilla.ForeColor
        btnUsuario.Text = IDUsuario & vbCrLf & "(" & Tipo & ")"
        btnUsuario.Margin = btnUsuarioPlantilla.Margin
        btnUsuario.TextAlign = btnUsuarioPlantilla.TextAlign
        btnUsuario.BackColor = btnUsuarioPlantilla.BackColor
        btnUsuario.Location = btnUsuarioPlantilla.Location
        btnUsuario.Cursor = btnUsuarioPlantilla.Cursor
        btnUsuario.Font = btnUsuarioPlantilla.Font
        btnUsuario.TabStop = False


        btnUsuario.BackgroundImage =
My.Resources.usuario_no_aprobado()
        If usuarioAprobado Then
            btnUsuario.BackgroundImage =
My.Resources.usuario_aprobado()
        End If
```

```
        btnUsuario.Tag = IDUsuario
        AddHandler btnUsuario.Click, AddressOf ClickVerUsuario

        btnEditar.Size = btnEditarPlantilla.Size
        btnEditar.FlatStyle = btnEditarPlantilla.FlatStyle
        btnEditar.ForeColor = btnEditarPlantilla.ForeColor
        btnEditar.Text = btnEditarPlantilla.Text
        btnEditar.BackColor = btnEditarPlantilla.BackColor
        btnEditar.Location = btnEditarPlantilla.Location
        btnEditar.Cursor = btnEditarPlantilla.Cursor
        btnEditar.TabStop = False

        btnEditar.Tag = IDUsuario
        AddHandler btnEditar.Click, AddressOf InterfazEditarUsuario

        btnEliminar.Size = btnEliminarPlantilla.Size
        btnEliminar.FlatStyle = btnEliminarPlantilla.FlatStyle
        btnEliminar.ForeColor = btnEliminarPlantilla.ForeColor
        btnEliminar.Text = btnEliminarPlantilla.Text
        btnEliminar.BackColor = btnEliminarPlantilla.BackColor
        btnEliminar.Location = btnEliminarPlantilla.Location
        btnEliminar.Cursor = btnEliminarPlantilla.Cursor
        btnEliminar.TabStop = False

        btnEliminar.Tag = {IDUsuario, btnUsuario.Text.Replace(vbCrLf,
" ")}
        AddHandler btnEliminar.Click, AddressOf EliminarUsuario

        pnlTemporal.Controls.Add(btnUsuario)
        If Not IDUsuario.Equals(miUsuario) Then
            pnlTemporal.Controls.Add(btnEliminar)
        End If
        pnlTemporal.Controls.Add(btnEditar)

        AddHandler pnlTemporal.MouseEnter, AddressOf fixScroll
        AddHandler pnlTemporal.MouseWheel, AddressOf fixScroll
        pnlUsuarios.Controls.Add(pnlTemporal)
        pnlUsuarios.Focus()

        totalUsuarios += 1
        lblCantidadUsuarios.Text = "(" + totalUsuarios.ToString() +
")"
    End Sub

    Private Sub fixScroll(sender As Object, e As Object)
        pnlUsuarios.Focus()
    End Sub

    Private Sub CheckDatosCorrectos(Optional sender As Object =
Nothing, Optional e As EventArgs = Nothing) Handles btnAgregar.Click
        If String.IsNullOrWhiteSpace(txtID.Text) Then
            MessageBox.Show("Debe ingresar un ID de usuario.",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
```

```
            Return
        End If

        If String.IsNullOrEmpty(txtContraseña.Text) Then
            MessageBox.Show("Debe ingresar una contraseña.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
            Return
        End If

        Usuario.ActualizarDB(Me)
        Usuario.ContUsuariosNoAprobados(Me.frmMain)
    End Sub

    Private Sub TipoUsuario_Cambiado(sender As Object, e As EventArgs)
Handles radFuncionario.CheckedChanged,
radAdministrador.CheckedChanged, radAdscripto.CheckedChanged
        If sender.Checked Then
            tipoSeleccionado = sender.Text
        End If
    End Sub

    Public Sub InterfazNuevoUsuario(Optional sender As Object =
Nothing, Optional e As EventArgs = Nothing) Handles
btnNuevoUsuario.Click, btnCancelar.Click
        ' Reinicia la interfaz
        lblNuevoUsuario.Text = "Nuevo usuario"
        btnAgregar.Text = "Agregar usuario"

        previsualizando = False
        btnAgregar.Visible = True
        btnCancelar.Visible = False
        btnNuevoUsuario.Visible = False
        txtID.Enabled = True
        txtContraseña.Enabled = True
        radFuncionario.Checked = True
        txtNombre.Enabled = True
        txtApellido.Enabled = True
        tipoSeleccionado = "Funcionario"
        RestablecerControles()
    End Sub

    Private Sub RestablecerControles()
        ' Limpia los valores de los controles
        chkHabilitado.Checked = False
        chkHabilitado.Enabled = True
        radAdministrador.Checked = False
        radFuncionario.Checked = True
        radAdscripto.Checked = False
        txtID.Text = ""
        txtContraseña.Text = ""
        txtNombre.Text = ""
        txtApellido.Text = ""
        tipoSeleccionado = "Funcionario"
        radAdministrador.Enabled = True
```

```
        radFuncionario.Enabled = True
        radAdscripto.Enabled = True
    End Sub

    Private Sub InterfazEditarUsuario(sender As Object, e As
EventArgs)
        ' Prepara la interfaz para editar el usuario
        lblNuevoUsuario.Text = "Editar usuario"
        btnNuevoUsuario.Visible = False
        btnAgregar.Text = "Confirmar cambios"
        btnAgregar.Visible = True
        btnNuevoUsuario.Visible = True
        btnCancelar.Visible = True
        txtContraseña.Enabled = True
        Usuario.CargarDatos(sender.Tag, Me)
        txtID.Enabled = False
        previsualizando = False
        txtNombre.Enabled = True
        txtApellido.Enabled = True

        radAdscripto.Enabled = False
        If Not sender.Tag.Equals(miUsuario) Then
            radAdministrador.Enabled = True
            radFuncionario.Enabled = True
            chkHabilitado.Enabled = True
        Else
            radAdministrador.Enabled = False
            radFuncionario.Enabled = False
            chkHabilitado.Enabled = False
        End If

        If tipoSeleccionado.Equals("Adscripto") Then
            radAdministrador.Enabled = False
            radFuncionario.Enabled = False
            radAdscripto.Enabled = True
            radAdscripto.Checked = True
        End If
    End Sub

    Private Sub ClickVerUsuario(sender As Object, e As EventArgs)
        ' Llama a la función que permite mostrar los datos del usuario
        InterfazVerUsuario(sender.Tag)
        previsualizando = True
    End Sub

    Private Sub InterfazVerUsuario(ID As String)
        ' Muestra los datos del usuario
        RestablecerControles()
        Usuario.CargarDatos(ID, Me)
        lblNuevoUsuario.Text = "Previsualizar usuario"
        btnNuevoUsuario.Visible = True
        btnCancelar.Visible = False
        btnAgregar.Visible = False
        txtID.Enabled = False
```

```
        txtContraseña.Enabled = False
        radAdministrador.Enabled = False
        radFuncionario.Enabled = False
        radAdscripto.Enabled = False
        txtNombre.Enabled = False
        txtApellido.Enabled = False
    End Sub

    Private Sub CuentaHabilitada_Cambiado(sender As Object, e As
EventArgs) Handles chkHabilitado.Click
        If previsualizando Then
            chkHabilitado.Checked = Not chkHabilitado.Checked
        End If
    End Sub

    Private Sub Check_Numero(t As Object, e As KeyEventArgs) Handles
txtID.KeyDown
        ' Al escribir un caracter que no sea número lo ignora.
        If e.KeyCode.Equals(Keys.Delete) Or
e.KeyCode.Equals(Keys.Back) Or e.KeyCode.Equals(Keys.Left) Or
e.KeyCode.Equals(Keys.Right) Or e.KeyCode.Equals(Keys.Tab) Then
            e.Handled = False
            Return
        End If

        If Not Char.IsDigit(Chr(e.KeyValue)) Then
            e.SuppressKeyPress = True
        End If
    End Sub

    Private Sub EliminarUsuario(sender As Object, e As EventArgs)
        ' Pregunta al usuario si quiere eliminar el usuario y de ser
correcto lo elimina
        Dim result As Integer = MessageBox.Show("¿Está seguro de que
desea eliminar el usuario '" + sender.Tag(1) + "'?", "Eliminar
usuario", MessageBoxButtons.YesNo, MessageBoxIcon.Question)
        If result = DialogResult.No Then
            Return
        End If

        Usuario.EliminarUsuario(sender.Tag(0), Me)
        Usuario.ContUsuariosNoAprobados(Me.frmMain)
    End Sub
End Class
```

9.3.8 frmDatosUsuario.vb

```
Public Class frmDatosUsuario

    Friend frmMain As frmMain

    Public Sub New(frmMain As frmMain)
        'inicia el programa, en caso de que sea invitado lo detecta
        InitializeComponent()
        Me.frmMain = frmMain
    End Sub

    Private Sub frmDatosUsuario_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
        lblNombreUsuario.Text = frmMain.NombreUsuario
        txtNombre.Focus()
    End Sub

    Private Sub btnCancelar_Click(sender As Object, e As EventArgs)
Handles btnCancelar.Click
        ' Al clickear salir, nos desloguea y muestra la ventana de
inicio
        frmIngresarRegistro.Show()
        frmIngresarRegistro.BringToFront()
        Me.Hide()
        Me.frmMain.Dispose()
    End Sub

    Private Sub checkDatos(Optional sender As Object = Nothing,
Optional e As EventArgs = Nothing) Handles btnAceptar.Click
        If String.IsNullOrWhiteSpace(txtNombre.Text) Then
            MessageBox.Show("Debe escribir un nombre.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
            Return
        End If

        If String.IsNullOrWhiteSpace(txtApellido.Text) Then
            MessageBox.Show("Debe escribir un apellido.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk)
            Return
        End If

        Usuario.EditarDatos(Me)
    End Sub

    Private Sub EnterClick(sender As System.Object, e As
System.Windows.Forms.KeyEventArgs) Handles txtNombre.KeyDown,
txtApellido.KeyDown
        If e.KeyCode.Equals(Keys.Enter) Then
            btnAceptar.PerformClick()
        End If
    End Sub
End Class
```

9.3.9 frmDia.vb

```
Public Class frmDia

    ' Crea un wdiget que simula ser un mini calendario con horas

    Dim totalHoras As Integer = 0
    Dim dia As String

    Public Sub New()
        InitializeComponent()
    End Sub

    Private Sub me_Paint(sender As Object, e As PaintEventArgs)
Handles Me.Paint
        ' Dibuja un borde
        e.Graphics.DrawRectangle(New Pen(Color.White, 2), New
Rectangle(New Point(1, 34), New Size(New Point(pnlDias.Width + 2,
pnlDias.Height + 2))))
    End Sub

    Public Sub agregarHora(horario As String, materia As String)
        ' Agrega una nueva hora a la lista de horas
        Dim labelHorario As New Label
        Dim labelMateria As New Label

        lblNoHayHoras.Visible = False

        labelHorario.AutoSize = True
        labelHorario.Padding = New Padding(0, 4, 0, 0)
        labelHorario.ForeColor = SystemColors.Control
        labelHorario.TextAlign = ContentAlignment.MiddleLeft
        labelHorario.Font = New Font("Microsoft Sans Serif", 12,
FontStyle.Bold)
        labelHorario.Text = horario

        labelMateria.AutoSize = True
        labelMateria.Padding = New Padding(0, 3, 0, 0)
        labelMateria.TextAlign = ContentAlignment.MiddleLeft
        labelMateria.ForeColor = SystemColors.AppWorkspace
        labelMateria.Font = New Font("Microsoft Sans Serif", 12)
        labelMateria.Text = materia

        pnlDias.RowStyles.Add(New RowStyle(SizeType.AutoSize))
        pnlDias.RowStyles.Add(New RowStyle(SizeType.AutoSize))

        pnlDias.Controls.Add(labelHorario, 0, totalHoras)
        pnlDias.Controls.Add(labelMateria, 1, totalHoras)

        totalHoras += 1
        Me.Invalidate()
    End Sub
```

```
    Public Sub limpiar()
        totalHoras = 0
        lblNoHayHoras.Visible = True
        pnlDias.Controls.Clear()
        pnlDias.RowStyles.Clear()
    End Sub

    Private Sub frmDia_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        lblDia.Text = Me.Name.ToString()
    End Sub
End Class
```

9.3.10 frmDialogoEspere.vb

```
Public Class frmDialogoEspere

End Class
```

9.3.11 frmEditarServidor.vb

```
Imports MySql.Data.MySqlClient

Public Class frmEditarServidor
    Dim conexionCorrecta As Boolean = True
    Dim userInicial, contraInicial, serverInicial, dbInicial
    Dim testeado = True

    Private Sub btnCancelar_Click(sender As Object, e As EventArgs)
Handles btnCancelar.Click
        Me.Dispose()
    End Sub

    Private Sub frmEditarServidor_Shown(sender As Object, e As
EventArgs) Handles Me.Shown
        serverInicial = GetSetting("Minerva", "BaseDeDatos",
"IP").ToString()
        userInicial = GetSetting("Minerva", "BaseDeDatos",
"Usuario").ToString()
        contraInicial = GetSetting("Minerva", "BaseDeDatos",
"Contraseña").ToString()
        dbInicial = GetSetting("Minerva", "BaseDeDatos",
"DB").ToString()

        txtDB.Text = dbInicial
        txtPasswd.Text = contraInicial
        txtUsuario.Text = userInicial
        txtServidor.Text = serverInicial
        btnProbar.PerformClick()
    End Sub

    Private Sub btnProbar_Click(sender As Object, e As EventArgs)
Handles btnProbar.Click
        Me.Enabled = False
        Me.Cursor = Cursors.WaitCursor
        Dim ventanaEspere As New frmDialogoEspere
        ventanaEspere.Show()
        Dim Conn As MySqlConnection
        Try
            Conn = New MySqlConnection("server=" & txtServidor.Text &
";uid=" & txtUsuario.Text & ";password=" & txtPasswd.Text &
";database=" & txtDB.Text & ";")
            Conn.Open()
            lblEstado1.ForeColor = Color.GreenYellow
            lblEstado2.ForeColor = Color.GreenYellow
            lblEstado2.Text = "Conexión establecida"
            conexionCorrecta = True
            Conn.Dispose()
        Catch ex As Exception
            conexionCorrecta = False
            lblEstado1.ForeColor = Color.Red
            lblEstado2.ForeColor = Color.Red
```

```
            lblEstado2.Text = "Error al establecer la conexión"
        End Try
        testeado = True
        Me.Enabled = True
        Me.Cursor = Cursors.Default
        ventanaEspere.Dispose()
        Me.BringToFront()
    End Sub

    Private Sub btnAceptar_Click(sender As Object, e As EventArgs)
Handles btnAceptar.Click
        If Not conexionCorrecta Or Not testeado Then
            MessageBox.Show("El programa debe ser capaz de establecer
la conexión a la base de datos para poder continuar." & vbCrLf &
"Recuerde que debe testear la conexión antes de presionar aceptar.",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Stop)
            Return
        End If
        SaveSetting("Minerva", "BaseDeDatos", "IP", txtServidor.Text)
        SaveSetting("Minerva", "BaseDeDatos", "Usuario",
txtUsuario.Text)
        SaveSetting("Minerva", "BaseDeDatos", "Contraseña",
txtPasswd.Text)
        SaveSetting("Minerva", "BaseDeDatos", "DB", txtDB.Text)
        frmIngresarRegistro.BringToFront()
        Me.Dispose()
    End Sub

    Private Sub _TextChanged(sender As Object, e As EventArgs) Handles
txtServidor.TextChanged, txtPasswd.TextChanged,
txtUsuario.TextChanged, txtDB.TextChanged
        testeado = False
        lblEstado1.ForeColor = Color.SkyBlue
        lblEstado2.ForeColor = Color.SkyBlue
        lblEstado2.Text = "Conexión sin probar"
    End Sub
End Class
```

9.3.12 frmHorariosExternos.vb

```
Imports System.IO
Imports System.Data
Imports System.Reflection
Imports iTextSharp.text
Imports iTextSharp.text.pdf

Public Class frmHorariosExternos

    Dim frmMain As frmMain
    Dim frmAdministrar As frmAdministrar
    Public Sub New(Optional frmMain As frmMain = Nothing, Optional
frmAdministrar As frmAdministrar = Nothing, Optional NombreDocente As
String = Nothing)

        ' Llamada necesaria para el diseñador.
        InitializeComponent()

        ' Agregue cualquier inicialización después de la llamada a
InitializeComponent().
        Me.frmMain = frmMain
        Me.frmAdministrar = frmAdministrar
        If Not IsNothing(NombreDocente) Then
            lblTitulo.Text = "Horarios de " & NombreDocente
            cboGrupo.Visible = False
        End If
    End Sub

    Private Sub btnAceptar_Click(sender As Object, e As EventArgs)
Handles btnAceptar.Click
        If Not IsNothing(frmMain) Then
            frmMain.BringToFront()
        ElseIf IsNothing(frmAdministrar) Then
            frmAdministrar.BringToFront()
        End If
        Me.Hide()
    End Sub

    Private Sub Grilla_Load(sender As Object, e As EventArgs) Handles
Grilla.Load
        Me.FormBorderStyle = Windows.Forms.FormBorderStyle.None
        Me.Location = New Point(0, 0)
        Me.Size = SystemInformation.PrimaryMonitorSize
    End Sub

    Private Sub cboGrupo_SelectedIndexChanged(sender As Object, e As
EventArgs) Handles cboGrupo.SelectedIndexChanged
        If Not IsNothing(frmMain) Then
            frmMain.cboGrupo.SelectedIndex =
frmMain.cboGrupo.FindStringExact(cboGrupo.Text)
            frmMain.copiarGrilla()
        End If
```

```
    End Sub

    Public Sub btnExportPDF_Click(sender As System.Object, e As
System.EventArgs) Handles btnGuardarPdf.Click
        sender.Enabled = False
        sender.BackgroundImage =
My.Resources.guardar_como_pdf_seleccionado()

        'Creating iTextSharp Table from the DataTable data
        Dim pdfTable As New PdfPTable(Grilla.dgvMaterias.ColumnCount)
        Dim intTblWidth() As Integer = {5, 15, 15, 15, 15, 15, 15}
        pdfTable.SetWidths(intTblWidth)
        pdfTable.DefaultCell.Padding = 5
        pdfTable.WidthPercentage = 100
        pdfTable.HorizontalAlignment = Element.ALIGN_CENTER
        pdfTable.DefaultCell.BorderWidth = 1
        pdfTable.DefaultCell.HorizontalAlignment =
Element.ALIGN_CENTER

        'Adding Header row
        For Each column As DataGridViewColumn In
Grilla.dgvMaterias.Columns
            Dim titulo As String = column.HeaderText
            If titulo.Equals("Hora de inicio") Then
                titulo = "#"
            End If
            Dim cell As New PdfPCell(New Phrase(titulo,
FontFactory.GetFont("Microsoft Sans Serif", 10)))
            cell.HorizontalAlignment = Element.ALIGN_CENTER
            cell.BackgroundColor = New BaseColor(240, 240, 240)
            pdfTable.AddCell(cell)
        Next

        'Adding DataRow
        For Each row As DataGridViewRow In Grilla.dgvMaterias.Rows
            For Each cell As DataGridViewCell In row.Cells
                Dim valor As String
                Try
                    valor = cell.Value.ToString()
                Catch ex As Exception
                    valor = ""
                End Try

                pdfTable.AddCell(New Phrase(valor,
FontFactory.GetFont("Microsoft Sans Serif", 10)))
            Next
        Next
        Dim path As String = Nothing
        If SaveFileDialog1.ShowDialog = Windows.Forms.DialogResult.OK
Then
            path = SaveFileDialog1.FileName
        End If
        If IsNothing(path) Then
```

```
            sender.BackgroundImage =
My.Resources.guardar_como_pdf_normal()
            sender.Enabled = True
            Return
        End If

        'Exporting to PDF
        Using stream As New FileStream(path, FileMode.Create)
            Dim pdfDoc As New Document(PageSize.A4, 10, 10, 10, 10)
            pdfDoc.SetPageSize(iTextSharp.text.PageSize.A4.Rotate())
            PdfWriter.GetInstance(pdfDoc, stream)
            pdfDoc.Open()
            pdfDoc.Open()
            Dim headerPdf
            If Not IsNothing(Me.frmAdministrar) Then
                headerPdf = New Paragraph(lblTitulo.Text & vbCrLf &
vbCrLf, FontFactory.GetFont("Courier", 25, BaseColor.BLACK))
            Else
                headerPdf = New Paragraph("Horarios del grupo " &
frmMain.cboGrupo.Text & vbCrLf & vbCrLf,
FontFactory.GetFont("Courier", 25, BaseColor.BLACK))
            End If
            pdfDoc.Add(headerPdf)
            pdfDoc.Add(pdfTable)
            pdfDoc.Close()
            stream.Close()
        End Using
        MessageBox.Show("Horarios guardados.", "Horarios guardados.",
MessageBoxButtons.OK, MessageBoxIcon.Information)
        sender.BackgroundImage =
My.Resources.guardar_como_pdf_normal()
        sender.Enabled = True
    End Sub

    Private Sub btnGuardarPdf_Enter(sender As Object, e As EventArgs)
Handles btnGuardarPdf.MouseEnter

        pnlAyudabtnGuardarPdf.Visible = True
        sender.BackgroundImage = My.Resources.guardar_como_pdf_hover()
    End Sub

    Private Sub btnGuardarPdf_Leave(sender As Object, e As EventArgs)
Handles btnGuardarPdf.MouseLeave

        pnlAyudabtnGuardarPdf.Visible = False
        sender.BackgroundImage =
My.Resources.guardar_como_pdf_normal()
    End Sub

    Private Sub btnAceptar_Enter(sender As Object, e As EventArgs)
Handles btnAceptar.MouseEnter
        pnlAyudabtnAceptar.Visible = True
        sender.BackgroundImage = My.Resources.unfullscreen_hover()
    End Sub
```

```
    Private Sub btnAceptar_Leave(sender As Object, e As EventArgs)
Handles btnAceptar.MouseLeave
        pnlAyudabtnAceptar.Visible = False
        sender.BackgroundImage = My.Resources.unfullscreen_normal()
    End Sub
End Class
```

9.3.13 frmIngresarRegistro.vb

```
Public Class frmIngresarRegistro

    Dim primeraVez As Boolean = True

    Private Sub btnIngresar_Click(sender As Object, e As EventArgs)
Handles btnIngresar.Click
        ' Al clickear ingresar mostrar login
        Dim login As New frmLogin()
        login.Show()
        Me.Hide()
    End Sub

    Private Sub btnInvitado_Click(sender As Object, e As EventArgs)
Handles btnInvitado.Click
        ' Al clickear invitado mostrar Minerva
        Dim programa As New frmMain(True)
        programa.Show()
        Me.Hide()
    End Sub

    Private Sub btnRegistro_Click(sender As Object, e As EventArgs)
Handles btnRegistro.Click
        ' Al clickear registro mostrar registro

        Dim registro As New frmRegistro()
        registro.Show()
        Me.Hide()
    End Sub

    Private Sub btnPreferencias_Enter(sender As Object, e As
EventArgs) Handles btnPreferencias.MouseEnter
        pnlAyudabtnPreferencias.Visible = True
        sender.BackgroundImage = My.Resources.preferencias_hover()
    End Sub

    Private Sub btnPreferencias_Leave(sender As Object, e As
EventArgs) Handles btnPreferencias.MouseLeave
        pnlAyudabtnPreferencias.Visible = False
        sender.BackgroundImage = My.Resources.preferencias_normal()
    End Sub

    Private Sub btnPreferencias_Click(sender As Object, e As
EventArgs) Handles btnPreferencias.Click
        Dim editarServidor As New frmEditarServidor()
        editarServidor.ShowDialog(Me)
    End Sub

    Private Sub frmIngresarRegistro_Load(sender As Object, e As
EventArgs) Handles MyBase.Shown
        If Not primeraVez Then
            Return
```

```
        End If
        primeraVez = True
        Me.Cursor = Cursors.WaitCursor
        Me.Enabled = False
        Dim ventanaEspere As New frmDialogoEspere
        ventanaEspere.Show()

        ' Datos por defecto de la base de datos
        If String.IsNullOrEmpty(GetSetting("Minerva", "BaseDeDatos",
"IP").ToString()) Then
            SaveSetting("Minerva", "BaseDeDatos", "IP", "localhost")
        End If
        If String.IsNullOrEmpty(GetSetting("Minerva", "BaseDeDatos",
"Usuario").ToString()) Then
            SaveSetting("Minerva", "BaseDeDatos", "Usuario",
"Minerva")
        End If
        If String.IsNullOrEmpty(GetSetting("Minerva", "BaseDeDatos",
"Contraseña").ToString()) Then
            SaveSetting("Minerva", "BaseDeDatos", "Contraseña",
"Minerva")
        End If
        If String.IsNullOrEmpty(GetSetting("Minerva", "BaseDeDatos",
"DB").ToString()) Then
            SaveSetting("Minerva", "BaseDeDatos", "DB", "Minerva")
        End If

        ' Comprueba la conexión.
        Dim conexion As New Conexion(True)
        conexion.Close()
        ventanaEspere.Dispose()
        Me.Cursor = Cursors.Default
        Me.Enabled = True
        Me.BringToFront()
    End Sub
End Class
```

9.3.14 frmLimpiarDB.vb

```
Public Class frmLimpiarDB

    Private Sub btnConfirmar_Click(sender As Object, e As EventArgs)
Handles btnConfirmar.Click
        Dim result As Integer = MessageBox.Show("¿Está seguro de que
desea realizar estos cambios?" & vbCrLf & vbCrLf & "Se recomienda
realizar una copia de seguridad (backup) de la base de datos, para
poder restaurlarla en caso de fallos.", "Alterar base datos",
MessageBoxButtons.YesNo, MessageBoxIcon.Question)
        If result = DialogResult.No Then
            Return
        End If
        Me.Dispose()
    End Sub
End Class
```

9.3.15 frmLogin.vb

```
Public Class frmLogin

    Friend estadoAnimacion As Boolean = False
    Friend administrador As Boolean = False

    Private Sub frmLogin_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
        timerAnimacion.Start()
    End Sub

    Private Sub frmLogin_FormClosed(sender As Object, e As
FormClosedEventArgs) Handles MyBase.FormClosed
        frmIngresarRegistro.Dispose()
    End Sub

    Private Sub btnCancelar_Click(sender As Object, e As EventArgs)
Handles btnCancelar.Click
        frmIngresarRegistro.Show()
        frmIngresarRegistro.BringToFront()
        Me.Dispose()
    End Sub

    Private Sub Login(sender As Object, e As EventArgs) Handles
btnEntrar.Click
        Me.Cursor = Cursors.WaitCursor
        Usuario.Login(Me)
    End Sub

    Private Sub timerAnimacion_Tick(sender As Object, e As EventArgs)
Handles timerAnimacion.Tick
        If estadoAnimacion Then
            lblDatosInc.ForeColor = Color.IndianRed
            imgWarning.BackgroundImage = My.Resources.warningRojo
        Else
            lblDatosInc.ForeColor = Color.White
            imgWarning.BackgroundImage = My.Resources.warningBlanco
        End If

        estadoAnimacion = Not estadoAnimacion
    End Sub

    Private Sub checkEscrito(sender As Object, e As EventArgs) Handles
txtCi.TextChanged, txtContraseña.TextChanged
        btnEntrar.Enabled = Not (String.IsNullOrWhiteSpace(txtCi.Text)
Or String.IsNullOrEmpty(txtContraseña.Text))

        pnlError.Visible = False
        lblIngreseUsuario.Visible =
String.IsNullOrWhiteSpace(txtCi.Text)
        lblIngreseContraseña.Visible =
String.IsNullOrEmpty(txtContraseña.Text)
```

```
    End Sub

    Private Sub lblIngreseUsuario_Click(sender As Object, e As
EventArgs) Handles lblIngreseUsuario.Click
        txtCi.Text = ""
        txtCi.Focus()
    End Sub

    Private Sub lblIngreseContraseña_Click(sender As Object, e As
EventArgs) Handles lblIngreseContraseña.Click
        txtContraseña.Text = ""
        txtContraseña.Focus()
    End Sub

    Private Sub EnterClick(sender As System.Object, e As
System.Windows.Forms.KeyEventArgs) Handles txtCi.KeyDown,
txtContraseña.KeyDown
        If e.KeyCode.Equals(Keys.Delete) Or
e.KeyCode.Equals(Keys.Back) Or e.KeyCode.Equals(Keys.Left) Or
e.KeyCode.Equals(Keys.Right) Or e.KeyCode.Equals(Keys.Tab) Then
            e.Handled = False
            Return
        End If

        If sender Is txtCi Then
            If Not Char.IsDigit(Chr(e.KeyValue)) Then
                e.SuppressKeyPress = True
            End If
        End If

        If e.KeyCode.Equals(Keys.Enter) Then
            btnEntrar.PerformClick()
            e.SuppressKeyPress = True
        End If
    End Sub
End Class
```

9.3.16 frmMain.vb

```
Public Class frmMain

    Friend NombreUsuario As String
    Friend TipoUsuario As String
    Friend cuentaInvitado As Boolean = True

    Friend TurnoElegido As ToolStripMenuItem = Nothing
    Friend CursoElegido As ToolStripMenuItem = Nothing

    Dim estadoAnimacion As Boolean = False
    Dim prevGrupo As String = ""
    Dim frmHorariosExternos As New frmHorariosExternos(Me)
    Dim filtrando As Boolean = False
    Dim vista As String = "Día"

    Public Sub New(Optional invitado As Boolean = False, Optional
usuario As String = Nothing, Optional tipoUsuario As String =
"Funcionario")
        InitializeComponent()

        Me.cuentaInvitado = invitado
        Me.TipoUsuario = tipoUsuario
        Me.NombreUsuario = usuario

        cboGrupo.SelectedIndex = 0
        FuncionesMinerva.CargarGrupos(Me)
        TimerBtnRefrescar.Enabled = True
    End Sub

    Private Sub frmMain_Closed(sender As Object, e As
FormClosedEventArgs) Handles MyBase.FormClosed
        frmIngresarRegistro.Dispose()
    End Sub

    Private Sub frmMain_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        If Not cuentaInvitado Then
            btnAdministrar.Visible = True
            imgLogoInvitado.Visible = False
            imgLogoUsuario.Visible = True
            alertaAprobacion.Visible = True
            lblUsuario.Text = "Bienvenido usuario."
        End If
        Me.WindowState = FormWindowState.Maximized

        Usuario.ContUsuariosNoAprobados(Me)
        FuncionesMinerva.CrearMenuFiltrado(Me)
    End Sub

    Private Sub frmMain_Shown(sender As Object, e As EventArgs)
Handles MyBase.Shown
```

```
        FuncionesMinerva.CargarNombre(Me)
    End Sub

    Private Sub AbrirAdministracion(sender As Object, e As EventArgs)
Handles btnAdministrar.Click, alertaAprobacion.Click
        Dim abrirusuario As Boolean = False

        If sender Is alertaAprobacion Then
            abrirusuario = True
        End If

        Dim administracion As New frmAdministrar(Me.TipoUsuario, Me,
abrirusuario)
        administracion.ShowDialog(Me)
    End Sub

    Private Sub GrupoCambiado(sender As Object, e As EventArgs)
Handles cboGrupo.SelectedIndexChanged
        btnRefrescarHorarios.Enabled = False
        btnRecargar.Enabled = False
        btnFullscreen.Visible = False
        btnGuardarPdf.Visible = False

        If cboGrupo.Text.Equals("Elija un grupo") Then
            lblSeleccioneGrupo.Visible = True
            lblSeleccioneGrupo.BringToFront()
            lblSeleccioneGrupo2.Visible = True
            lblSeleccioneGrupo2.BringToFront()
            TimerBtnRefrescar.Enabled = True
            Return
        End If

        lblNomGrupo.Text = cboGrupo.Text.Trim()
        lblSeleccioneGrupo.Visible = cboGrupo.SelectedIndex = -1
        lblSeleccioneGrupo2.Visible = cboGrupo.SelectedIndex = -1

        If vista.Equals("Semana") Then
            Grilla.Visible = True
            Grilla.BringToFront()
            tblDias.Visible = False
            btnFullscreen.Visible = True
            btnGuardarPdf.Visible = True
        Else
            Grilla.Visible = False
            tblDias.Visible = True
            tblDias.BringToFront()
        End If

        TimerBtnRefrescar.Enabled = True

        If Not cboGrupo.Text.Equals(prevGrupo) Then
            Lunes.limpiar()
            Martes.limpiar()
            Miércoles.limpiar()
```

```
            Jueves.limpiar()
            Viernes.limpiar()
            Sábado.limpiar()
            Grilla.dgvMaterias.Rows.Clear()
            FuncionesMinerva.CargarInfoGrupo(Me)
            FuncionesMinerva.CargarHorariosGrupo(Me)
        End If
        prevGrupo = cboGrupo.Text
        imgLogoMAITs.Focus()
    End Sub

    Private Sub Salir(sender As Object, e As EventArgs) Handles
btnSalir.Click
        frmIngresarRegistro.Show()
        Me.Dispose()
    End Sub

    Public Sub RefrescarHorarios(Optional sender As Object = Nothing,
Optional e As EventArgs = Nothing) Handles btnRefrescarHorarios.Click,
btnRecargar.Click
        prevGrupo = "-1"
        RecargarGrupo()
    End Sub

    Private Sub TimerBtnRefrescar_Tick(sender As Object, e As
EventArgs) Handles TimerBtnRefrescar.Tick
        btnRefrescarHorarios.Enabled = True
        btnRecargar.Enabled = True
        TimerBtnRefrescar.Enabled = False
    End Sub

    Public Sub RecargarGrupo()
        Dim grupo As String

        grupo = cboGrupo.Text
        FuncionesMinerva.CargarGrupos(Me)
        cboGrupo.SelectedIndex = 0

        If Not (cboGrupo.FindStringExact(grupo) = -1) Then
            cboGrupo.SelectedIndex = cboGrupo.FindStringExact(grupo)
        End If
    End Sub

    Private Sub ClickVistaDias(sender As Object, e As EventArgs)
Handles btnVistaDias.Click
        vista = "Día"

        btnVistaDias.Enabled = False
        btnVistaDias.BackgroundImage = My.Resources.dia_seleccionado()

        btnVistaSemana.Enabled = True
        btnVistaSemana.BackgroundImage = My.Resources.semana_normal()

        RecargarGrupo()
```

```
        pnlAyudabtnVistaDias.Visible = False
    End Sub

    Private Sub ClickVistaSemana(sender As Object, e As EventArgs)
Handles btnVistaSemana.Click
        vista = "Semana"

        btnVistaSemana.Enabled = False
        btnVistaSemana.BackgroundImage =
My.Resources.semana_seleccionado()

        btnVistaDias.Enabled = True
        btnVistaDias.BackgroundImage = My.Resources.dia_normal()

        RecargarGrupo()
        pnlAyudabtnVistaSemana.Visible = False
    End Sub

    Private Sub VerHorariosEnPantallaCompleta(sender As Object, e As
EventArgs) Handles btnFullscreen.Click
        sender.BackgroundImage = My.Resources.fullscreen_normal()
        frmHorariosExternos.cboGrupo.Items.Clear()

        For i = 1 To cboGrupo.Items.Count - 1
            frmHorariosExternos.cboGrupo.Items.Add(cboGrupo.Items(i))
        Next

        frmHorariosExternos.cboGrupo.SelectedIndex =
cboGrupo.SelectedIndex - 1
        copiarGrilla()
        frmHorariosExternos.ShowDialog(Me)
    End Sub

    Private Sub GuardarPdf(sender As Object, e As EventArgs) Handles
btnGuardarPdf.Click
        copiarGrilla()
        frmHorariosExternos.btnExportPDF_Click(sender, e)
    End Sub

    Public Sub copiarGrilla()
        Dim targetRows = New List(Of DataGridViewRow)
        For Each sourceRow As DataGridViewRow In
Grilla.dgvMaterias.Rows
            If (Not sourceRow.IsNewRow) Then
                Dim targetRow = CType(sourceRow.Clone(),
DataGridViewRow)
                For Each cell As DataGridViewCell In sourceRow.Cells
                    targetRow.Cells(cell.ColumnIndex).Value =
cell.Value
                Next

                targetRows.Add(targetRow)
            End If
        Next
```

```
        frmHorariosExternos.Grilla.dgvMaterias.Columns.Clear()
        For Each column As DataGridViewColumn In
Grilla.dgvMaterias.Columns

frmHorariosExternos.Grilla.dgvMaterias.Columns.Add(CType(column.Clone(
), DataGridViewColumn))
        Next

frmHorariosExternos.Grilla.dgvMaterias.Rows.AddRange(targetRows.ToArra
y())
    End Sub

    Private Sub fixScroll(sender As Object, e As EventArgs) Handles
tblMaterias.MouseWheel, tblMaterias.MouseEnter
        pnlMaterias.Focus()
    End Sub

    Public Sub FiltroTurnoCambiado(sender As Object, e As EventArgs)
        If Not IsNothing(TurnoElegido) Then
            If sender Is TurnoElegido Then
                sender.checked = False
                TurnoElegido = Nothing
                CheckFiltrando()
                RecargarGrupo()
                Return
            End If
            TurnoElegido.Checked = False
        End If

        sender.checked = True
        TurnoElegido = sender
        CheckFiltrando()
        RecargarGrupo()
    End Sub

    Public Sub FiltroCursoCambiado(sender As Object, e As EventArgs)
        If Not IsNothing(CursoElegido) Then
            If sender Is CursoElegido Then
                sender.checked = False
                CursoElegido = Nothing
                CheckFiltrando()
                RecargarGrupo()
                Return
            End If
            CursoElegido.Checked = False
        End If

        sender.checked = True
        CursoElegido = sender
        CheckFiltrando()
        RecargarGrupo()
    End Sub

    Private Sub CheckFiltrando()
```

```
        filtrando = True
        btnFiltrar.BackgroundImage = My.Resources.filtrar_click()
        If CursoElegido Is Nothing And TurnoElegido Is Nothing Then
            filtrando = False
            btnFiltrar.BackgroundImage = My.Resources.filtrar_normal()
        End If
    End Sub

    Private Sub Animacion_1_E(sender As Object, e As EventArgs)
Handles btnGuardarPdf.MouseEnter
        If Not sender.Enabled Then
            Return
        End If

        pnlAyudabtnGuardarPdf.Visible = True
        sender.BackgroundImage = My.Resources.guardar_como_pdf_hover()
    End Sub

    Private Sub Animacion_1_L(sender As Object, e As EventArgs)
Handles btnGuardarPdf.MouseLeave
        If Not sender.Enabled Then
            Return
        End If

        pnlAyudabtnGuardarPdf.Visible = False
        sender.BackgroundImage =
My.Resources.guardar_como_pdf_normal()
    End Sub

    Private Sub Animacion_2_E(sender As Object, e As EventArgs)
Handles btnRefrescarHorarios.MouseEnter
        pnlAyudabtnRefrescarHorarios.Visible = True
        sender.BackgroundImage = My.Resources.refrescar_hover()
    End Sub

    Private Sub Animacion_2_L(sender As Object, e As EventArgs)
Handles btnRefrescarHorarios.MouseLeave
        pnlAyudabtnRefrescarHorarios.Visible = False
        sender.BackgroundImage = My.Resources.refrescar_normal()
    End Sub

    Private Sub Animacion_3_L(sender As Object, e As EventArgs)
Handles btnRecargar.MouseLeave
        sender.BackgroundImage = My.Resources.refrescar_normal()
        pnlAyudabtnRecargar.Visible = False
    End Sub

    Private Sub Animacion_3_E(sender As Object, e As EventArgs)
Handles btnRecargar.MouseEnter
        sender.BackgroundImage = My.Resources.refrescar_hover()
        ' al entrar a el botón btnAgregarAsignatura cambiar la imagen
        pnlAyudabtnRecargar.Visible = True
    End Sub
```

```
    Private Sub Animacion_4_E(sender As Object, e As EventArgs)
Handles btnVistaDias.MouseEnter
        If Not sender.enabled Then
            Return
        End If

        sender.BackgroundImage = My.Resources.dia_hover()
        pnlAyudabtnVistaDias.Visible = True
    End Sub

    Private Sub Animacion_4_L(sender As Object, e As EventArgs)
Handles btnVistaDias.MouseLeave
        If Not sender.enabled Then
            Return
        End If

        sender.BackgroundImage = My.Resources.dia_normal()
        pnlAyudabtnVistaDias.Visible = False
    End Sub

    Private Sub Animacion_5_E(sender As Object, e As EventArgs)
Handles btnVistaSemana.MouseEnter
        If Not sender.enabled Then
            Return
        End If

        sender.BackgroundImage = My.Resources.semana_hover()
        pnlAyudabtnVistaSemana.Visible = True
    End Sub

    Private Sub Animacion_5_L(sender As Object, e As EventArgs)
Handles btnVistaSemana.MouseLeave
        If Not sender.enabled Then
            Return
        End If

        sender.BackgroundImage = My.Resources.semana_normal()
        pnlAyudabtnVistaSemana.Visible = False
    End Sub

    Private Sub Animacion_6_E(sender As Object, e As EventArgs)
Handles btnFullscreen.MouseEnter
        If Not sender.enabled Then
            Return
        End If

        sender.BackgroundImage = My.Resources.fullscreen_hover()
        pnlAyudabtnFullscreen.Visible = True
    End Sub

    Private Sub Animacion_6_L(sender As Object, e As EventArgs)
Handles btnFullscreen.MouseLeave
        If Not sender.enabled Then
            Return
```

```
        End If

        sender.BackgroundImage = My.Resources.fullscreen_normal()
        pnlAyudabtnFullscreen.Visible = False
    End Sub

    Private Sub Animacion_7_L(sender As Object, e As EventArgs)
Handles btnFiltrar.MouseLeave
        lblFiltrado.Text = "Filtrado de grupos (activo)"
        sender.backgroundimage = My.Resources.filtrar_click()

        If Not filtrando Then
            sender.BackgroundImage = My.Resources.filtrar_normal()
            lblFiltrado.Text = "Filtrado de grupos (inactivo)"
        End If

        pnlAyudabtnFiltrar.Visible = False
    End Sub

    Private Sub Animacion_7_E(sender As Object, e As EventArgs)
Handles btnFiltrar.MouseEnter
        lblFiltrado.Text = "Filtrado de grupos (activo)"
        sender.backgroundimage = My.Resources.filtrar_click()

        If Not filtrando Then
            sender.BackgroundImage = My.Resources.filtrar_hover()
            lblFiltrado.Text = "Filtrado de grupos (inactivo)"
        End If

        pnlAyudabtnFiltrar.Visible = True
    End Sub

    Private Sub MenuFiltrado(sender As Object, e As MouseEventArgs)
Handles btnFiltrar.Click
        cmsFiltrado.Show(sender, New Point(e.X, e.Y))
    End Sub

    Private Sub Animacion_8_E(sender As Object, e As EventArgs)
Handles alertaAprobacion.MouseEnter
        pnlAyudaalertaAprobacion.Visible = True
    End Sub

    Private Sub Animacion_8_L(sender As Object, e As EventArgs)
Handles alertaAprobacion.MouseLeave
        pnlAyudaalertaAprobacion.Visible = False
    End Sub

    Private Sub imgLogoInvitado_Click(sender As Object, e As
EventArgs) Handles imgLogoInvitado.Click, imgLogoUsuario.Click
        Dim frmAcerca As New frmAcerca()
        frmAcerca.ShowDialog(Me)
    End Sub
End Class
```

9.3.17 frmRegistro.vb

```
Public Class frmRegistro

    Dim estadoAnimacion As Boolean = False
    Private Sub Registro(sender As Object, e As EventArgs) Handles
btnEntrar.Click
        Usuario.Registro(Me)
    End Sub

    Private Sub frmLogin_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
        timerAnimacion.Start()
        Dim cantidadAdministradores As Integer =
PersistenciaUsuarios.CantAdministradores()
        If cantidadAdministradores <= 0 Then
            radAdministrador.Checked = True
            radFuncionario.Enabled = False
            radAdscripto.Enabled = False
        End If
    End Sub

    Private Sub frmLogin_FormClosed(sender As Object, e As
FormClosedEventArgs) Handles MyBase.FormClosed
        frmIngresarRegistro.Dispose()
    End Sub

    Private Sub btnCancelar_Click(sender As Object, e As EventArgs)
Handles btnCancelar.Click
        frmIngresarRegistro.Show()
        frmIngresarRegistro.BringToFront()
        Me.Dispose()
    End Sub

    Private Sub timerAnimacion_Tick(sender As Object, e As EventArgs)
Handles timerAnimacion.Tick
        If estadoAnimacion Then
            lblDatosInc.ForeColor = Color.IndianRed
            imgWarning.BackgroundImage = My.Resources.warningRojo
        Else
            lblDatosInc.ForeColor = Color.White
            imgWarning.BackgroundImage = My.Resources.warningBlanco
        End If

        estadoAnimacion = Not estadoAnimacion
    End Sub

    Private Sub checkEscrito(sender As Object, e As EventArgs) Handles
txtCi.TextChanged, txtContraseña.TextChanged
        btnEntrar.Enabled = Not (String.IsNullOrWhiteSpace(txtCi.Text)
Or String.IsNullOrEmpty(txtContraseña.Text))

        pnlError.Visible = False
```

```
        lblIngreseUsuario.Visible =
String.IsNullOrWhiteSpace(txtCi.Text)
        lblIngreseContraseña.Visible =
String.IsNullOrEmpty(txtContraseña.Text)
    End Sub

    Private Sub lblIngreseUsuario_Click(sender As Object, e As
EventArgs) Handles lblIngreseUsuario.Click
        txtCi.Text = ""
        txtCi.Focus()
    End Sub

    Private Sub lblIngreseContraseña_Click(sender As Object, e As
EventArgs) Handles lblIngreseContraseña.Click
        txtContraseña.Text = ""
        txtContraseña.Focus()
    End Sub

    Private Sub EnterClick(sender As System.Object, e As
System.Windows.Forms.KeyEventArgs) Handles txtCi.KeyDown,
txtContraseña.KeyDown
        If e.KeyCode.Equals(Keys.Delete) Or
e.KeyCode.Equals(Keys.Back) Or e.KeyCode.Equals(Keys.Left) Or
e.KeyCode.Equals(Keys.Right) Or e.KeyCode.Equals(Keys.Tab) Then
            e.Handled = False
            Return
        End If

        If sender Is txtCi Then
            If Not Char.IsDigit(Chr(e.KeyValue)) Then
                e.SuppressKeyPress = True
            End If
        End If
        If e.KeyCode.Equals(Keys.Enter) Then
            btnEntrar.PerformClick()
            e.SuppressKeyPress = True
        End If
    End Sub
End Class
```

9.3.18 frmVistaGrilla.vb

```
Public Class frmVistaGrilla
    Private Sub dgvMaterias_CellContentClick(sender As Object, e As
EventArgs) Handles dgvMaterias.SelectionChanged
        sender.ClearSelection()
    End Sub
End Class
```

10. Herramientas utilizadas

10.1 Visual Studio

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta múltiples lenguajes de programación tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby, PHP

10.2 Microsoft Word

Microsoft Word es una aplicación informática orientada al procesamiento de textos. Fue creado por la empresa Microsoft, y viene integrado en el paquete ofimático denominado Microsoft Office.

10.3 MySQL Workbench

MySQL Workbench es una herramienta visual de diseño de bases de datos que integra desarrollo de software, administración de bases de datos, diseño de bases de datos, creación y mantenimiento para el sistema de base de datos MySQL.

10.4 Lucidchart

Lucidchart es una herramienta de diagramación basada en la web, que permite a los usuarios colaborar y trabajar juntos en tiempo real, creando diagramas de flujo, organigramas, esquemas de sitios web, diseños UML, mapas mentales, prototipos de software y muchos otros tipos de diagrama.

10.5 Dropbox

Dropbox es un servicio de alojamiento de archivos multiplataforma en la nube, operado por la compañía Dropbox. El servicio permite a los usuarios almacenar y sincronizar archivos en línea y entre ordenadores y compartir archivos y carpetas con otros usuarios y con tabletas y móviles.

10.6 Photoshop

Adobe Photoshop es un editor de gráficos rasterizados desarrollado por Adobe Systems Incorporated. Usado principalmente para el retoque de fotografías y gráficos.

10.7 Inkscape

Inkscape es un editor de gráficos vectoriales gratuito y de código libre. Inkscape puede crear y editar diagramas, líneas, gráficos, logotipos, e ilustraciones complejas.

10.8 Google Drive

Google Drive es un servicio de alojamiento de archivos. Cada usuario cuenta con 15 gigabytes de espacio gratuito para almacenar sus archivos, ampliables mediante diferentes planes de pago. Es accesible a través del sitio web desde computadoras y dispone de aplicaciones para Android e iOS que permiten editar documentos y hojas de cálculo.

10.9 Edmodo

Edmodo es una plataforma tecnológica, social, educativa y gratuita, que permite la comunicación entre los alumnos y los profesores en un entorno cerrado y privado a modo de microblogging, creado para un uso específico en educación.

10.10 TeamViewer

TeamViewer es un software cuya función es conectarse remotamente a otro equipo. Entre sus funciones están: compartir y controlar escritorios, reuniones en línea, videoconferencias y transferencia de archivos entre ordenadores.

10.11 Adobe Acrobat Reader

Adobe Acrobat es una familia de programas o aplicaciones informáticas desarrollados por Adobe Systems, diseñados para visualizar, crear y modificar archivos con el formato Portable Document Format, más conocido como PDF. El uso del formato PDF es muy común para mostrar texto con un diseño visual ordenado.

10.12 Skype

Skype es un software que permite comunicaciones de texto, voz y vídeo sobre Internet