PROYECTO FINAL DE SEGUNDO AÑO

CARRERA DE ANALISTA DE SISTEMAS
ANALISTA PROGRAMADOR WEB .NET
2023

Generalidades

- La entrega final se hará durante la última clase de tutoría. Durante las clases de tutorías deberán realizarse entregas parciales de acuerdo a lo establecido en la sección Tutorías & Entregas.
- Se deberá realizar la entrega mediante correo electrónico, del software junto con toda la documentación exigida en la sección ***Requerimientos de Entrega***:
  - **Asunto**: Entrega Proyecto Final ANP 2023
  - **Destinatarios**: segundoanalista@bios.edu.uy y tutor asignado (OBLIGATORIO)
  - **Compartido por Drive**: documentación y solución completa
  - **Cuerpo**: Nombre y Cedula de los integrantes del grupo que realiza la entrega
- Se deberán formar grupos de 1 a 3 personas, **los cuales deberán inscribirse en bedelía desde el día 20 de noviembre hasta el día 01 de diciembre**. ***No se aceptarán inscripciones fuera de fecha***. Luego de esto, Bedelía ***publicara*** la asignación de tutores, así como de horarios. ***La inscripción es únicamente por vía mail, con el siguiente formato***:
  a) Asunto: *Inscripción a Proyecto Final Segundo Año*
  b) Contenido:
    a. Cedula y Nombre de ***todos*** los integrantes del grupo que se presenta
    b. Franja Horaria a la que se concurre a clase
- La asistencia a la última tutoría es obligatoria para **todos** los miembros del grupo ya que se realizará la defensa en máquina del proyecto.

Idea General

Se desea realizar el desarrollo de un sistema para ***AeropuertosAmericanos*** (empresa que gestiona aeropuertos, vuelos y venta de pasajes para América)

Los usuarios empleados (que gestionan los aeropuertos, los vuelos y la venta de pasajes) deberán tener usuario de logueo (***único*** en el Sistema), contraseña, nombre completo y el tipo de labor asignada (Gerente – Vendedor – Admin). De los clientes (que son los que compran pasajes) se deberá saber el número de pasaporte, el nombre, número de tarjeta de crédito (con la cual pagan los pasajes) y la contraseña para loguearse

Los conceptos que forman parte de la realidad se describen en cada punto en donde se deben usar, para generar las funcionalidades del sistema propuesto

Arquitectura Solicitada

- Para el acceso público y del cliente, se deberá generar un Sitio Web ASP.Net, que trabajará con Entity Framework con Modelo DBF.
- Para el acceso de los empleados, se deberá generar una Aplicación Web MVC, que trabajará con 3 capas (Entidades, Lógica, Persistencia).
- La única base de datos estará instalada en un servidor de datos.
- Ambas UI Web deberán estar instaladas en un servidor IIS

Funcionalidades Mínimas del Sitio Web ASP .Net

Formulario Web: Principal (formulario por defecto del sitio)
Actor Participante: público
Resumen:

a) En este formulario se desplegarán las partidas y arribos de un aeropuerto seleccionado por el usuario. Los aeropuertos podrán ser seleccionados a partir de un control *DropDownList*. Las partidas y arribos deberán desplegarse en *grillas*. En el caso de las partidas se deberá desplegar los vuelos que aún ***no partieron***, ordenados por fecha y hora de partida, y desplegando: Fecha/hora partida, destino (aeropuerto de llegada del vuelo) indicando Ciudad y País del mismo, y cantidad de pasajes vendidos. En el caso de los arribos, se deberán desplegar los vuelos que aún ***no llegaron***, ordenados por fecha y hora de llegada, y desplegando: fecha/hora llegada, y aeropuerto del cual proviene (aeropuerto de partida del vuelo) indicando Ciudad y País del mismo, y cantidad de pasajes vendidos. Se podrá filtrar por un rango de fechas (para ambas listas). ***Uso obligatorio de Linq para obtener datos y realizar filtros***

b) El formulario deberá tener acceso directo al formulario de logueo al sistema de clientes

Formulario Web: Logueo cliente
Actor Participante: público
Resumen: Mediante este formulario se permitirá el logueo de un cliente al sitio (ingreso de pasaporte y contraseña). Si el cliente se autentica correctamente, el sistema lo re direccionará al ***Formulario Histórico de Compras***.

Formulario Web: Histórico de Compras
Actor Participante: Cliente
Resumen: Mediante ese formulario, un cliente tendrá acceso al histórico de sus compras de pasajes. Se deberá desplegar en una grilla, ordenados por la fecha de la compra**: Fecha – Vuelo – Monto de la compra – Empleado que generó la compra**. En caso que se seleccione una compra de pasaje, se deberá desplegar la información completa (incluyendo aeropuerto de partida (con ciudad y país), aeropuerto de llegada (con ciudad y país) y la lista de asientos/clientes para los cuales se compra pasaje. ***Uso obligatorio de Linq para obtener datos y realizar filtros***

Funcionalidades Mínimas de la Aplicación Web MVC

Logueo Empleado
Actor Participante: público
Resumen: Mediante esta vista se permitirá el logueo de un usuario empleado al sitio (ingreso de usuario y contraseña). Si el empleado se autentica correctamente, el sistema lo redireccionará a una ***vista de bienvenida que cuente con el menú principal del sistema***. Luego de logueado el usuario**, *todos los accesos a la base de datos, deberán ser a través de su propio usuario de Sql Server (no podrán realizarse acciones con acceso de usuarios públicos)***

ABM de Ciudades
Actor Participante: Empleado
Resumen: Permite realizar alta, baja, modificación y consulta individual de Ciudades. Se deberá saber un código de 6 letras (identificador único en el sistema), nombre de la ciudad y país en la cual se encuentra ubicada. Ejemplos: MVDURU (Montevideo, Uruguay), RIOBRA (Rio de Janeiro, Brasil). Considerar que en una misma ciudad podrán ubicarse más de un aeropuerto. No se pueden eliminar ciudades que tengan aeropuertos asociados. ***En el listado general se deberá poder filtrar por parte del nombre***.

ABM Aeropuertos
Actor Participante: Empleado
Resumen: Permite realizar alta, baja, modificación y consulta individual de aeropuertos. Se deberá saber: Código (identificador único en el sistema) compuesto por 3 letras, nombre, dirección, en cual ciudad esta, monto del impuesto que se cobra por pasajero si embarca en un vuelo (impuestos partida) y monto del impuesto que se cobra por pasajero si desciende de un vuelo (impuesto llegada). Los impuestos pueden ser de valor 0 (no se cobran) Ejemplo: MVD (Aeropuerto de Carrasco, Avenida de las Américas XXX, Montevideo/Uruguay, 100, 0). No se pueden eliminar aeropuertos que tengan vuelos asociados. ***En el listado general se deberá poder filtrar por parte del nombre***.

ABM de Clientes
Actor Participante: Empleado
Resumen: Permite realizar alta, baja, modificación y consulta individual de clientes. Se deberá saber: Numero de Pasaporte (identificador único en el sistema), nombre, número de tarjeta con la cual realizará los pagos de los pasajes, y contraseña de acceso al sistema. No se pueden eliminar clientes que tengan pasajes o ventas asociadas. ***En el listado general se deberá poder filtrar por parte del nombre***.

Alta Vuelos
Actor Participante: Empleado
Resumen: Permite realizar el alta de un vuelo. De cada vuelo se deberá saber: código (identificador único y generado por el sistema) alfanumérico de 15 de largo, fecha/hora y aeropuerto de partida, fecha/hora y aeropuerto de llegada, precio del pasaje, y cantidad de asientos (mínimo 100, máximo 300). El código del vuelo lo deberá generar el sistema considerando la fecha y hora de partida y el aeropuerto. Deberá tener el siguiente formato: **yyyyMMddHHmmAer** (202401202050MVD). *Los aeropuertos deberán poderse seleccionar desde una lista desplegable*.

Venta de Pasajes
Actor Participante: Empleado
Resumen: Permite realizar la venta de un pasaje para un vuelo. El usuario deberá seleccionar el Vuelo para el cual se vende el pasaje y el Cliente que lo paga (ambos datos deberán ser seleccionados desde listas desplegables). Además, se deberá almacenar el número de venta (identificador único y autogenerado por el sistema), la fecha de la compra (automática del sistema), y el precio total de la venta (deberá ser la suma del precio de los pasajes del vuelo, más los impuestos que cobran los aeropuertos de partida y llegada de dicho vuelo – calculo automático). Luego de definir todos estos datos básicos, deberán seleccionarse los asientos que se compran para ese vuelo (número de asiento - cliente que usara dicho pasaje). Cualquier cliente puede comparar pasajes para cualquier cliente. Solo un cliente quedara definido como el que se encarga de pagar la compra. Además, se deberá saber cuál empleado genero la venta. Deberá validarse que aún quedan lugares disponibles en el vuelo para realizar la venta.

Listado Interactivo de Vuelos
Actor Participante: Empleado
Resumen: Se desplegarán todos los vuelos del sistema, ordenados por fecha de salida. Si se seleccionada un vuelo, deberán desplegarse todos los pasajes vendidos (asiento – cliente pasajero). Se podrán realizar filtros por:

a. vuelos que ya partieron o que aún no lo han hecho (se considera la fecha/hora de salida para tales efectos).
b. vuelos que parten de un determinado aeropuerto (seleccionado de una lista desplegable)
c. vuelos que parten determinada fecha
d. limpiar filtros, y volver a desplegar la lista inicial completa

Uso obligatorio de Linq para realizar filtros

Requerimientos de Implementación

- Implementación completa del sistema con tecnologías .NET en lenguaje C#. Obligatorio la entrega en **.Net Framework 4.5.2 o 4.6**.
- La información deberá almacenarse obligatoriamente en una base de datos **SQL Server.**
- El script de la base de datos debe generarse ***manualmente***, sin la ayuda de un asistente. Deberá contener el ***Esquema de creación de la base de datos***, ***Stored Procedures*** necesarios para realizar todas las tareas solicitadas, ***Creación de usuarios y permisos*** solicitados y necesarios, e ***Inserción*** de datos de prueba. Las restricciones sobre datos, deberán realizarse en la propia estructura de la tabla (uso de los modificadores ***unique***, ***check***, ***default***).
- Los empleados en esta versión no tendrán mantenimiento a través del sistema, se deben generar a nivel de base de datos al menos 10 empleado.
- Las eliminaciones a nivel de la base de datos deberán ser físicas o lógicas en función de las posibilidades. Es decir, si un elemento a eliminar no tiene elementos dependientes dentro de los registros de la base de datos, o estos se pueden eliminar, se elimina físicamente; de lo contrario se hará una eliminación lógica. ***Obligatorio de implementar***
- Para el desarrollo de la Aplicación Web MVC, se debe utilizar la arquitectura en 3 capas vista en el curso, mediante la utilización de bibliotecas de clases. Se deberán crear los proyectos correspondientes a las 3 bibliotecas y luego referenciarlas desde la aplicación. ***Obligatorio de implementar***
- Para el desarrollo del Sitio Web ASP, se deberá utilizar Entity Framework DBF. Para ello se deberá generar una biblioteca para el modelo de Entity Framework y luego referenciarla desde el sitio. ***Obligatorio de implementar***
- ***Obligatorio*** el uso de clases para la comunicación de datos entre componentes (tanto para invocación como respuesta).
- Los componentes de Lógica y Persistencia deberán ser generados en base a los patrones de ***Fábrica y Singleton*** vistos en clase. No deberá implementarse ningún otro patrón.
- Siempre se deberá de trabajar en forma conectada (ADO.NET) para el manejo de la información de la base de datos en el caso de la capa de ***Persistencia***.
- Los componentes deberán lanzar excepciones en caso de error. No se contempla otra forma de comunicación de errores entre componentes

Requerimientos de Entrega

- Modelo Conceptual.
- MER
- Diagrama de Clases completo de la Arquitectura en capas (incluye a todos los componentes, clases, interfaces y relaciones entre ellos)
- Solución completa del Software (2 proyectos de UI – 4 bibliotecas de clases)
- Script de la base de datos

Nota: todos los diagramas deberán ser generados con una herramienta para lenguaje UML. Estos deberán ser entregados en forma digital: una copia del archivo original del diagrama y una copia en formato PDF o JPG

Tutorías y Entregas

A continuación, se detallan las entregas sugeridas:

- Primera Sesión (*semana del 29/01 al 02/02*):
  - Modelo Conceptual
  - DER
  - Script BD completa (tablas – procedimientos – manejo de usuarios – baja lógica)
  - Biblioteca para el modelo de Entity Framework completa
  - Biblioteca Entidades Compartidas completa
  - Biblioteca Persistencia completa

- Segunda Sesión (*semana del 19/02 al 23/02*):
  - Biblioteca Lógica completa
  - UI: Sitio Web de ASP.Net completo
  - UI: aplicación web MVC (general y logueo)

- Tercera Sesión (*semana del 11/03 al 15/03*):
  - Entrega de todo lo solicitado en el punto ***Requerimientos de Entrega***.
  - ***Defensa obligatoria del proyecto***. Para ello el grupo deberá concurrir antes de su horario de tutoría, para realizar la instalación del sistema. Dicha instalación deberá respetar la arquitectura propuesta.