Trabajo práctico de Gestión de Datos

GRUPO: 44

INTEGRANTES:

- Ortiz Emiliano
- Veliz Ezequiel

TEMA: Estrategia de desarrollo de TP

¿Cómo lo dividimos?

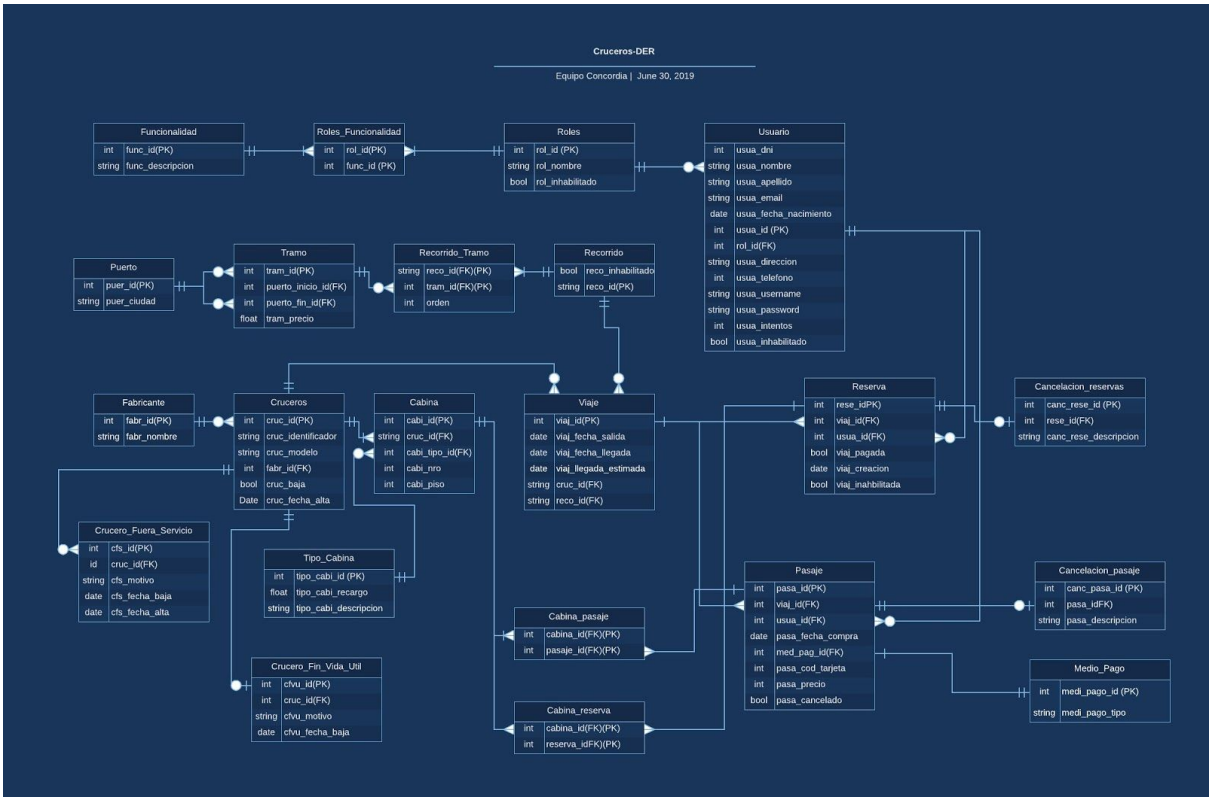
Esto lo hicimos juntos

Para facilitar la creación del DER(el cual adjuntaremos en uno de los puntos a continuación) utilizamos la herramienta colaborativa LucidChart, la cual nos permitió trabajar en un mismo archivo en remoto.
Luego de decidir qué parte realizaría cada uno creamos un repositorio en GitHub y comenzamos a trabajar cada uno en su rama hasta que logramos terminarlo y lo integramos.

Esto lo dividimos

Emiliano	**Ezequiel**
ABM Crucero	ABM Recorrido
Compra/Reserva Pasaje	ABM Rol
Generación de Viaje	Listado Estadístico
Migración	Pago Reserva

DER(Diagrama Entidad Relación)



DER en LucidChart:
https://www.lucidchart.com/invitations/accept/fd0ce2dd-7241-458a-9ecb-5d43d64de7e3

Inicio

Esta sección es nuestro punto de entrada al programa y está compuesta por las siguientes tres partes:

Bienvenida

Aquí se decidió por colocar 2 opciones para nuestros dos tipos de usuarios principales, el cliente(que solo puede generar reservas y comprar pasajes),el cual no necesita de un sistema de autenticación, y los usuarios regulares(con rol distinto a cliente) que poseen otros roles, como administrador por ejemplo, y que necesitan de un sistema de autenticación ya que manejan información más sensible, como la alta y baja de cruceros, roles etc.
Si se selecciona la opción Cliente, se crea una instancia genérica(solo con el id del rol) de la clase Usuario y se procede al Menú principal. En cambio, si se selecciona la opción Usuario se procede a la funcionalidad del Login.

Login

Para acceder a las funcionalidades del usuario se codificó un login que con dos input de texto. Los cuales son username y password.
Una vez completado los campos y apretado el botón logear internamente hace la validación que los campos estén completos. Luego llama a un stored procedure el cual le pasa estos datos completados.
El stored procedure internamente busca el usuario y la cantidad de intentos que tiene.
Si no existe el usuario carga el resultado de retorno con -1
Si existe el usuario hacer lo siguiente:

- Convierte el pass ingresado por el usuario a SHA2_256
- Busca el pass del usuario en la base de datos
- Y los compara si son distintas

Si son distintas carga el retorno del procedure con la cantidad de intentos restantes y hace el update de intentos en la base de datos .
Si son iguales y los intentos restantes son distintos de 0 cargo en el retorno un número 4 y hago un update del número de intentos que le quedan al usuario con 3.
Luego en la aplicación compra el resultado y dependiendo el caso muestra los intentos restantes, dice que el usuario no existe o ingresa al menú principal creando una instancia de usuario con el correspondiente rol. En el caso de que se haya inhabilitado el rol del usuario, este tendrá un NULL en la fila correspondiente, en ese caso, le asignamos un rol_id de 0, y cuando volvemos a la vista del Login verificamos si es 0, y si lo es sencillamente mostramos un error en el que indicamos que el usuario no posee un rol válido.
En el login agregamos la funcionalidad que al ingresar el usuario se revisen todas las reservas y se verifiquen que no haya reservas mayores a 3 días y en caso que se encuentren se cancelen estas modificando el valor en la tabla por inhabilitada.

Usuarios administradores

Para suplantar el ABM usuario agregamos 3 usuarios administradores.
Sus datos son:

usuario: admin pass: w23e	usuario: admin1 pass: w23e	usuario: admin2 pass: w23e

Menú Principal

Esta vista requiere como parámetros a su vista anterior(para volver a ella en caso de que esta se cierre) y a una instancia de Usuario con su respectivo rol para poder hallar sus funcionalidades.
Ahora se crea una List de KeyValuePair con el id como key y la descripción de la funcionalidad como value según las funcionalidades que posea el rol del usuario. Esta List se agrega a un ListBox en el que el usuario puede seleccionar la funcionalidad que desea

realizar, luego presiona en el botón Seleccionar y esta ventana se cierra y se abre el ABM correspondiente.
En el momento que se está cargando esta vista se ejecuta el método verificarReservas() de la clase Database, este método es el encargado de verificar y dar de baja todas las reservas que posean una antigüedad mayor a tres días(lo cual nos habilita para darlas de baja). Elegimos este lugar para ejecutarla ya que es un punto en común por el que pasan todos los que van a utilizar el sistema, tanto usuarios como clientes, por ser esta ventana, el punto en el que deben elegir la funcionalidad a ejecutar.

ABM Crucero

Se creo un menu el cual te permite seleccionar funcionalidad que queres usar o volver atrás al menú principal.

Alta crucero

Para el alta de crucero se codificó un formulario el cual puedas ingresar todos los datos necesarios para la creación de un crucero (Identificador, Modelo, Su empresa dueña, fecha de alta) ,además la posibilidad de agregar las cabinas que posee dicho crucero (con sus respectivos datos: nro de habitación, piso y su tipo de cabina) el cual es controlado que se ingresen solo datos válidos, además la posibilidad de visualizar las cabinas agregadas mediante un data grid view. Se puede controlar la cantidad de cabinas agregadas mediante textBox bloqueado en los datos del crucero que se actualiza automáticamente.
Una vez finalizada la carga de datos se encuentra en la parte de abajo del formulario el botón que permite crear el crucero que actualiza los datos en la base de datos y también se encuentra un botón limpiar y volver hacia atrás en caso que quieras borrar todos los datos ingresados en el crucero.
Internamente el botón “crear” verifica que todos los campos indispensables estén completados y procede a hacer lo siguiente:

- LLama al procedure insertarCrucero que verifica que el identificador no esté repetido. En caso que no exista lo inserta en la tabla cruceros con todos sus datos y devuelve el id. Y en caso que exista devuelve -1 para que la aplicación pueda informarle al usuario.
- Luego si se llama al procedure insertarCabina que se encarga insertar todas las cabinas del crucero con sus respectivos datos en la tabla cabina.
- Si se creó todo con éxito se le informa al usuario que fue cargado con exito.

Baja crucero

Para la baja del crucero hay que entender que hay 4 tipos de bajas:

1. **La baja definitiva con cancelación de pasajes:** En la cual se da debaja el crucero de manera lógica y se le agrega en la tabla de cruceros dados de baja definitiva y se cancelan todos los pasajes

2. **La baja definitiva con reemplazo de crucero para los pasajes:** En la cual se da de baja al crucero pero todos sus viajes se los reemplaza con un crucero de las mismas características y disponibilidad
3. **Baja por fuera de servicio y cancelación de pasajes:** que se da de baja un crucero por cierto tiempo
4. **Baja por fuera de servicio y desplazamiento de pasajes:** que mueve los pasajes con ese crucero cierto tiempo.

Baja de cruceros tiene un buscador de cruceros que te permite buscar por id, modelo o su empresa dueña y los resultados que van a ser siempre cruceros habilitados van a aparecer en un data grid view, el cual una vez seleccionada la fila del crucero que quieras elegir y aprietes dar de baja te va a aparecer una nueva pestaña que te dejará seleccionar el tipo de baja que quieras darle.
La búsqueda se realiza a través de una query que se autogenera en la aplicación y se guarda en un objeto SqlCommand y se ejecuta devolviendo la tabla que se carga en el data grid view.
La nueva pestaña baja contará con los siguientes campos, dos radiobox si es una baja definitiva o una fuera de servicio. Al accionarlo va a deshabilitar los campos de una o la otra de acuerdo cual. Si es una baja definitiva y queres cancelar los pasajes tenes checkbox que te permite seleccionarlo y textbox para explicar el motivo.
Si en cambio queres dar de baja el crucero y reemplazar los pasajes tienes que tener destildado el checkbox cancelar pasaje y apretar el botón baja.
En el caso que sea fuera de servicio puedes seleccionar el checkbox dar de baja pasajes y al igual que el anterior tienes que dar el motivo en cambio si esta destildado vas a desplazar los pasajes tantos días como días fuera de servicio este (Los días se van a calcular automáticamente con la fecha ingresada).
A continuación prosigo a explicar la forma de proceder para cada caso:

La baja definitiva con cancelación de pasajes:

Utiliza un procedure que setea el campo inhabilitado en la tabla crucero y agrega al crucero a la tabla crucero_fin_vida_util.
Si se pudo hacer ejecuta otro procedure que se encarga de cancelar todos los pasajes de ese crucero desde la fecha que se hace hacia adelante, seteando el campo de inhabilitado al pasaje y agregandolos a la tabla cancelacion_pasajes con el motivo.

La baja definitiva con reemplazo de crucero para los pasajes:

Uilizo un procedure que se encarga de buscar el id del crucero reemplazante.En este procedure creo 4 tablas temporales. 1 para los cruceros que no cumple con la disponibilidad horaria, el 2 para los viajes al cual hay que reemplazar , el 3 para las cantidades de cada tipo de cabina que dispone el crucero al que hay que reemplazar , la 4 para los cruceros que cumplen con la disponibilidad de fechas. SI no hay ningún viaje a reemplazar devuelve 0 el procedure. Si hay viajes hago un while por cada viaje del crucero a reemplazar (en la tabla 2) y busco los cruceros que no cumplen para ese viaje y los agrego a la tabla 1. Una vez terminado tomó los crucero que sí cumplen con la disponibilidad de fechas por una query que busca los ids que no están en la tabla 1 (cruceros no reemplazantes) y los agrego a la tabla 4(cruceros

que cumplen con la disponibilidad de fechas). Luego comparó la tabla 4 (cruceros que cumplen con la disponibilidad de fechas) con mi tabla 3 (cabinas de cada tipo) y veo si tiene la misma cantidad de cada tipo de cabina o un número mayor. En caso que no se encuentre nada devuelve -1.
Utiliza un procedure que setea el campo inhabilitado en la tabla crucero y agrega al crucero a la tabla crucero_fin_vida_util.
Y mediante un procedure de reemplazo de crucero remplazo el crucero en cada pasaje y ademas cambio la cabina que figura en el pasaje por una cabina del otro crucero.
Al finalizar avisa al usuario el id del crucero que lo reemplazó.

Baja por fuera de servicio y cancelación de pasajes:
En esta opción doy de baja el crucero mediante un procedure que se llama FueraServicioCrucero que se encarga de agregarlo a la tabla de cruceros_fuera_de_servicio y setea como inhabilitado el crucero.
Luego llama a un procedure que es cancelarPasajes que cancela todos los pasajes de la fecha actual hasta la fecha dada de baja.
Y se le informa al usuario la cantidad de pasajes dados de baja.

Baja por fuera de servicio y desplazamiento de pasajes:
En esta opción doy de baja el crucero mediante un procedure que se llama FueraServicioCrucero que se encarga de agregarlo a la tabla de cruceros_fuera_de_servicio y setea como inhabilitado el crucero.
Luego se llama a un procedure que es DesplazamientoDePasajes que le hace un update a todos los pasajes de la fecha actual en adelante con la fecha en la que sería el viaje sumandole los días de desplazamiento.
Se le informa al usuario la cantidad de días de desplazamiento.

Modificación crucero

Esta funcionalidad permite dar de modificar los datos del crucero. Su primera pantalla es la de búsqueda de cruceros mediante id, modelo y marca.
La búsqueda en modificación funciona de una forma muy similar a la de baja salvo que te va a traer todos los cruceros que cumplan sin importar que esté habilitado o no.
Una vez seleccionado una fila del data grid view y apretando el botón modificar, van a aparecer todos los datos cargados del crucero en la pantalla. donde vas a poder modificar si esta inhabilitado o no, su marca y su fecha de alta. También van a aparecer el id , el identificador y modelo pero estos campos van a estar bloqueados para no modificar la integridad de la tabla y asumimos que un crucero no va a poder cambiar de nombre y de modelo durante su vida útil.
Una vez apretado el botón modificar va a validar que los campos estén completos y va a llamar a un procedure llamado ModificarCrucero que se encargará de hacer el update de los datos y luego se avisará al usuario si fue existo.

ABM Puerto

Esta era una de las dos vistas que no debían ser codificadas, aunque igualmente decidimos formar una tabla compuesta por dos campos, puer_id para identificarla y puer_ciudad para saber donde estaba emplazado el mismo, para representar a esta entidad ya que debíamos utilizarla para poder crear la entidad tramo, la cual es una conexión entre dos puertos, y es la base para formar un recorrido.

ABM Recorrido

Antes que nada, un recorrido está compuesto por cuatro entidades básicas, siendo estas:

1. Puerto, entidad descrita anteriormente
2. Tramo, entidad formada por un tram_id para identificarla, un puer_id_inicio para indicar el inicio de un tramo, un puer_id_destino para indicar el destino del mismo(estas últimas dos siendo FK apuntando a puerto) y un precio
3. La entidad recorrido en sí misma, la cual está compuesta por un id, un campo booleano en el cual indicaremos si el mismo está inhabilitado o no y el campo reco_codViejo, el cual muestra el código que poseían los recorridos de la tabla maestra, los cuales migramos pero de todas maneras no nos pareció necesario seguir utilizando ya que lo reemplazamos por una PK autoincremental, además nos pareció poco práctico que un usuario tenga que recordar un código alfanumérico para cada recorrido ya que a simple vista, los mismo no te comunican ningún detalle del mismo.
4. Finalmente la entidad recorrido_tramo la cual está compuesta por su PK, formada por las FKs reco_id(PK de recorrido) y tram_id(PK de tramo), de esta forma formamos una relación inequívoca e irrepetible entre tramos y recorridos y finalmente el campo orden con el cual podemos mantener los distintos tramos ordenados según como fueron cargados, ya que si hay más de un tramo, no se podría saber dónde comienza y donde finaliza el recorrido

Ahora, las tres secciones principales de este ABM a las cuales se puede acceder desde una botonera vertical, la cual posee un botón para cada una y uno extra para volver al menú principal y seleccionar otro ABM:

Alta recorrido

En este ABM podemos crear un recorrido nuevo y lo podemos hacer creando una lista de tramos, para agregar tramos creamos dos opciones:

1. Crear un tramo que previamente no existía , seleccionando un puerto de destino, uno de inicio y un precio y agregarlo a la lista

2. Seleccionar un tramo de los ya existentes(buscando por puerto) y agregarlo a la lista

En ambos casos, se crea una instancia de la clase Tramo y se agrega a una lista de tramos en el abm de alta de recorrido, la diferencia es que a los tramos no persistidos se les agrega un flag que indica que no lo están para poder diferenciarlos y poder persistirlos al presionar guardar recorrido. En los dos casos anteriores, tanto para crear o seleccionar un tramo anterior, se debían seleccionar puertos, el enunciado indicaba que se debía desplegar una lista en caso de que necesitáramos seleccionar un puerto, pero preferimos armar un pequeño buscador de puertos ya que armar un listado con casi 50 opciones de puertos para seleccionar nos pareció demasiado extenso y engorroso para el usuario.
A modo de filtro, antes de agregar cualquier tramo a la lista, se verifica que no se hayan agregado con anterioridad a la misma lista, otros tramos con las mismas características, es decir, se verifica que no se hayan agregado tramos que posean tanto el mismo puerto de incio como el mismo puerto de destino, ya que esto generaría un error de constraint al momento de persistir la relación tramo recorrido.
Una vez que se tiene la lista de tramos(la cual se puede observar en un DataGridView) completa, se presiona en guardar para persistir el recorrido, lo primero que se hace es verificar que haya algún tramo seleccionado, luego se persisten los tramos no persistidos(en esta función se verifica que el tramo no esté persistido ya, si lo está se devuelve su id, sino se persiste y se devuelve su id), luego se persiste una nueva entrada en la tabla recorrido y se devuelve su id, al tener el id del recorrido y la lista de tramos se pasa a persistir la relación recorrido_tramo, para el orden del tramo se utiliza el índice que tiene el tramo en la lista de tramos.

Baja recorrido

En esta sección utilizamos dos opciones para buscar un recorrido:

1. Buscar el recorrido por su puerto de inicio
2. Buscar el recorrido por su puerto de destino

Ambos utilizan el buscador de puertos descrito anteriormente. Se puede buscar el recorrido completando mínimamente uno de los dos puertos o ambos en caso de que se requiera.
Una vez que se presiona buscar se verifica que al menos uno de los dos puertos este completo y se procede a buscar algún recorrido que cumpla con los requisitos.
Una vez concluida la búsqueda se pueden observar los resultados en un DataGridView, desde donde se pueden ver los detalles del recorrido(todos los tramos con sus respectivos puertos) para ver si es el buscado a través de la opción en un botón, o también darlo de baja, mediante otro botón en la fila.

Modificación recorrido

Esta funcionalidad comienza con una ventana idéntica a la de baja en la que se utiliza el mismo procedimiento para buscar el recorrido deseado, una vez buscado el recorrido se presiona sobre el botón Modificar de la fila y se procede a una ventana similar a la de alta, con la diferencia de que en esta se puede inhabilitar o no un recorrido, en el caso de que se quite alguno de los tramos del recorrido, estos se agregan a una lista de recorridos por

eliminar, en el caso de que se agregue algún recorrido este agrega a una lista de recorridos por agregar, en el caso de que el tramo ya se encuentre en la lista, este no se volverá a agregar, y en el caso de que se cree un tramo con puertos idénticos pero precio distinto, este tampoco se agregara, ya que no tiene sentido tener dos tramos iguales con distinto precio, en cualquier caso habría que tener un ABM tramos para modificar el precio del mismo para esto pero no fue pedido en el enunciado.
Una vez que se presiona guardar se verifica que haya mínimamente un tramo en el recorrido, luego si cumple, se itera la lista de tramos por quitar y se elimina la relacion recorrido tramo correspondiente. Luego de esto se itera la lista de tramos por agregar y se agregan las relaciones recorrido tramo que sean necesarias, al terminar estos dos últimos pasos se debe reconstruir el orden de los tramos, ya que al quitar o agregar tramos pueden quedar orden ilógicos como por ejemplo 1, 2, 1, por lo que se llama a un pequeño store procedure(de nombre ActualizarOrden) que se encarga, primero de obtener la cantidad total de tramos que posee el recorrido, luego se crea una pequeña tabla formada por dos columnas, el id del tramo y el orden que ocupa en la tabla recorrido_tramo(esto último se obtiene utilizando la función row_number, la cual me devuelve el número de fila del campo dado), luego llega el momento de realizar un update sobre la tabla recorrido_tramo(en la que actualizo el orden) joineada con la tabla generada anteriormente sobre la columna tram_id y con un where en el que verifico que pertenezcan al recorrido pasado por parametro. Para finalizar obtenemos el valor de la checkbox que me indica si el recorrido está habilitado o no y realizamos un update sobre el mismo para asentar este valor.
Luego de todo, se obtiene la lista de recorridos de la vista anterior y se actualizan todos los datos pertinentes, para que en caso de que el usuario guarde los datos, vuelva a la vista anterior y sin limpiar, vuelva a seleccionar el mismo recorrido, obtenga los datos más recientes.

ABM Rol

Primero, la relación Rol está compuesta por 4 entidades:

1. La entidad Rol, la cual posee 3 campos, una id, un nombre, como puede ser “Administrador”, o “Cliente” y finalmente un campo booleano que indica si el mismo está inhabilitado o no. Un rol puede poseer muchas funcionalidades, por lo que se crean dos entidades más, Funcionalidad y Rol_Funcionalidad.
2. La entidad Funcionalidad está compuesta por 2 campos, su id y una descripción que me dice a que me habilita acceder la misma, como por ejemplo ABM Crucero, ABM Usuario, etc.
3. la entidad Rol_Funcionalidad que representa una relación de muchos a muchos entre rol y funcionalidad(un rol puede poseer muchas funcionalidades y una funcionalidad puede pertenecer a distintos roles), esta está compuesta solo por su PK, la cual es compuesta ya que está formada por el rol_id(FK que apunta a Rol) y por funcionaliad_id(FK que apunta a Funcionalidad).
4. Finalmente la entidad Usuario, que entre otros campos posee el campo rol_id, de esta manera logramos asignarle un rol al mismo.

Alta Rol

En esta vista cargamos todos las funcionalidades disponibles en la base de datos en un ComboBox y le indicamos al usuario que debe seleccionar una o varias, a medida que las va seleccionando las mismas se agregan a un DataGridView(siempre verificando que la misma no esté ya presente) en el cual tiene la opción de quitar alguna si así lo desea, además de esto le pedimos un nombre para el rol, como por ejemplo “Administrador”.
Una vez que está todo listo, el usuario presiona en el botón guardar, se verifica que haya ingresado algún nombre, que el DataGridView posea al menos una funcionalidad, que no exista ya un rol en la base de datos con el mismo nombre y que no exista un rol con las mismas funcionalidades, en cualquiera de estos casos se le avisa al usuario, si pasa todas las validaciones se persiste la entidad rol y se retorna su id, luego de esto se persiste la entidad rol_funcionalidad utilizando el id del rol agregado y los ids de las distintas funcionalidades agregadas a la lisa.

Baja Rol

Esta funcionalidad lista los roles disponibles(los habilitados) en un ListBox, luego uno simplemente selecciona uno y presiona el botón Inhabilitar. En el interior, esta funcion , ademas de inhabilitar al rol, le quita la asociación a todas las personas que lo posean utilizando un pequeño store procedure llamado InhabilitarUsuarios al que se le pasa por parámetro el rol_id y setea en null el rol de todos los usuarios con ese rol.

Modificación Rol

Esta funcionalidad lista a todos los roles(tanto habilitados como inhabilitados), luego de esto el usuario debe seleccionar el que desea editar, y al seleccionarlo se completan todos los campos para poder ser editados, el nombre en un TextBox y el listado de funcionalidades en un DataGridView, en la fila de la funcionalidad el usuario puede presionar el botón Quitar y se remueve esa funcionalidad del rol, también posee un ComboBox para poder agregar funcionalidades que antes no poseía, finalmente sobre el final hay un checkbox en el que el usuario puede modificar la habilitación del rol, luego de presionar el botón Guardar se generan 2 listas de Funcionalidades, una para persistir(las agregadas) y otra para remover la relación preexistente de rol_funcionalidad.

ABM Usuario

Este era el segundo de los ABMs que no había que codificar, aunque igualmente hubo que diseñar la entidad usuario ya que era parte fundamental del trabajo.
La entidad usuario está compuesta por la PK campo _id, los valores de datos, dni, nombre, apellido, email, fecha de nacimiento, dirección y teléfono cada uno en su respectivo campo, una FK que apunta a rol(rol_id), tal como fue explicado en ABM Rol, un campo username, un campo password(el cual está encriptado con SHA256), y un campo intentos, este último siendo un campo incremental que va aumentando cada vez que el usuario introduce su contraseña mal, una vez que la introduce bien, se setea en 0 con un UPDATE, en caso de que se introduzca mal 3 veces o más, el usuario queda bloqueado y no se puede loguear.

Generación de Viaje

Para la creacion de viaje diseñamos un formulario que te permite ingresar la fecha de inicio, fecha de finalización, un crucero y un recorrido.
Se le permite seleccionar al usuario una fecha de inicio mayor a la fecha actual (Asumimos nosotros que no va a ser necesario crear un viaje con una fecha menor a la actual). Y se permite seleccionar una fecha de finalización no menor a la fecha de inicio (Ya que no tendría lógica para el negocio que no se valide). Para seleccionar el crucero se abre una ventana que te permite buscar cruceros por id y modelo mediante un select que se genera en el propia aplicación y ejecutándola en la base de datos(Esta opción la elegimos por facilidad a la hora de editar el query con los parámetros y que el componente de selección fue hecho de forma genérica), es importante agregar que solo se podrán seleccionar cruceros habilitados y una vez seleccionada se valida que no viaje en esas fechas. Para seleccionar el recorrido pasa algo muy similar a el de cruceros, la query es generada en la aplicación y te permite buscar un recorrido por su id y por su código en la tabla maestra. Una vez finalizada la carga de datos y seleccionado el botón crear se llama a el procedure generarViaje que valida que el crucero esté libre si lo esta genera el viaje y le informa al usuario que se creó con éxito en caso contrario se le informa al que el crucero está ocupado.

Compra/Reserva Pasaje

Compra/Reserva consta de 3 partes:

- Buscado de viaje
- Compra viaje
- Reserva viaje

Buscador de viaje

Buscador de viaje permite encontrar viajes por su fecha de inicio, puerto de inicio y puerto de fin. una vez seleccionado estos ítems y seleccionar el botón buscar. LLamara a un store procedure que se encargará de buscar los viajes disponibles en esas fechas y buscará las cabinas libres de esos viajes filtrando las cabinas que están vendidas y/o reservadas. Aparecerán todos los viajes con las cabinas disponibles por medio de un data grid view. Al seleccionar uno o varios viajes con sus cabinas(las filas) se puede optar por comprar o reservar y se le pasara una instancia de los datos seleccionados a los distintos componentes.

Compra viaje

Al abrirse la instancia de compra aparecerá un formulario para completar los datos del comprador(Asumimos que se va a generar un pasaje con todas las cabinas correspondientes). Mediante el evento leav del textBox cada vez que se saque el foco del textbox del dni se ejecutará un procedure de búsqueda de usuarios llamado datosUsuario que buscara en la tabla usuario la coincidencia del dni (Para esta búsqueda tomamos

siempre el primer elemento devuelve la búsqueda ya que en realidad el dni no se podría repetir, salvo por el caso de la migración de datos y al ser un caso excepcional optamos por top 1 de devolución). Con los datos devueltos cargamos los datos a los text box.
Cuando aparece una coincidencia con el dni cargamos una variable con que los datos salieron de la base de datos y generado el pasaje se hace el update de estos datos por medio de procedure llamado updateUsuario.
En caso de que el usuario haya cargado los datos y no los hayan traído de la base de datos se procede a hacer un insert del mismo en la tabla usuarios con el rol de cliente.
Después hay que cargar los medio de pago que se puede elegir entre efectivo y tarjeta de débito el cual se valida dependiendo cual elija el usuario si se completan determinados campos o no.
Dejamos la posibilidad a futuro de compra en cuotas con tarjeta de crédito agregando el campo cantidad de cuotas en medio de pago.
Una vez completado todo los campos y seleccionando el botón comprar este validará los campos , hará el update del usuario o insert según corresponda y explicado más arriba, luego generar la compra mediante el procedure CrearPasaje y luego se le asocian todas las cabinas con el pasaje comprado en la tabla cabinas_pasaje.
Una vez finalizado el proceso se le informará al usuario las cabinas compradas el número de pasaje y el número de viaje y los puertos de inicio y fin.

Reserva viaje

La parte de búsqueda de datos de usuario funciona exactamente igual a compra viaje llamando al mismo store procedure.
Una vez finalizado la carga de datos del usuario el sistema generará una reserva a nombre del usuario con un procedure llamado crearReserva que inserta los datos necesarios en la tabla reserva y luego llama un procedure llamado cargarCabinaReserva que asocia un número de reserva con una cabina.
Una vez finalizado el se le informa al usuario su número de reserva, el viaje, el puerto de inicio y finalización del viaje.

Tanto compra como reserva validan que el pasajero no tenga el viajeComprado o reservado desde antes y que no tenga otro viaje en esas fechas.

Listado Estadístico

Esta funcionalidad le permite generar estadísticas muy básicas sobre la cantidad de ventas de un recorrido dado, la cantidad de cabinas de un recorrido y los días fuera de servicio de un crucero, para hacerlo más segmentado se encuentra la opción de seleccionar un semestre y año dado. Para obetener los años, realizo un SELECT MIN(pasa_fecha_compra) FROM …., con lo que obtengo el año más pequeño con el que empezar a calcular las estadísticas, más temprano no tendría sentido ya que no habría datos y agrego los años suficientes para llegar hasta el actual, y para los semestres, completo con los dos posibles. Una cosa en común que poseen las tres estadísticas que describire a continuacion es que todas utilizan la función DATEPART(), tanto como para

verificar que pertenezcan al año seleccionado como para verificar que pertenecen al semestre seleccionado, esto se logra averiguando si pertenecen al primer o segundo trimestre(para el caso del primer semestre) o verificando que pertenezcan al tercer o cuarto trimestre(para el caso del segundo semestre) utilizando el parámetro QUARTER, a continuación la descripción de las estadísticas:

Recorridos con más pasajes comprados

En este utilizo las tablas recorrido, viaje y pasaje, en el SELECT, selecciono el id de recorrido y un count(*) para contar la cantidad de filas por recorrido(utilizando un GROUP BY(reco_id) puedo lograr esto, sino dará error), y en el WHERE igualo el recorrido.reco_id con el viaje.reco_id, además igualo el viaje.viaj_id con el pasaje.viaj_id
De esta manera obtengo todos los pasajes vendidos de todos los viajes que tengan un recorrido dado, finalmente lo ordeno de forma descendente por la columna COUNT(*) y utilizando un TOP 5 obtengo los 5 recorridos más relevantes.
Esta información se muestra en un DataGridView en tres columnas, una para mostrar el id del recorrido, otra para mostrar la cantidad de pasajes y finalmente una columna con un botón para ver los detalles del recorrido.
Si se presiona en el botón para ver los detalles, se abrirá una nueva ventana, en la que se mostrará:

1. Los tramos del recorrido.
2. El precio total del recorrido
3. Un detalle de cuantas ventas de pasajes se realizaron para cada mes del semestre

El detalle de las ventas por mes se obtiene con una query muy similar a la anterior, la diferencia radica en que en el WHERE agrego que recorrido.reco_id debe ser igual al recorrido seleccionado, en que en el select utilizo un CASE WHEN DATEPART(MONTH, fecha de pasaje) para filtrar las ventas por mes, luego agrupo las filas por mes y utilizando un COUNT(*) cuento la cantidad total por mes.

Recorridos con más cabinas libre en cada uno de sus viajes

Esta funcionalidad es un poco más compleja que las otras dos, por lo que en nuestras pruebas tardó un par de segundos en mostrar los resultados solicitados.
Utilizamos un Select principal que itera a través de toda la tabla recorridos y en cada iteración utilizo la siguiente subquery:

- Esta subquery relaciona las tablas recorrido, viaje, crucero, y cabina, con el id del recorrido externo para de esta manera obtener todas las cabinas que poseían todos los cruceros que realizaron un viaje con un recorrido dado, pero verificando que las mismas no se encuentren en la tabla cabina_pasaje(entidad que relaciona cada pasaje con la/s cabinas seleccionadas con el mismo), como resultado de esto se obtienen todas las cabinas posibles de todos los recorridos, excluyendo a aquellas que hayan sido ocupadas en el periodo dado.

Luego de esto, agrupo por el reco_id(id del recorrido), ordeno de forma descendente y selecciono las primeras cinco. Al igual que la funcionalidad anterior, muestro los resultados en tres columnas.
Al presionar el botón de Detalle, se abre una ventana en la que se pueden ver:

1. Los tramos del recorrido.
2. El precio total del mismo.
3. Una tabla con dos columnas en la que se muestra la cantidad de cabinas libres por mes.

Para ver el detalle realice un procedimiento distinto al utilizado en las otras dos funcionalidades.
En vez de utilizar una única query con un case when, utilice 6 queries, donde cada una devuelve la cantidad de cabinas libres para un recorrido dado para un mes dado de los seis del semestre, luego aplique un unión entre estas 6 queries y como resultado obtuve una sola tabla con la cantidad de cabinas libres desagregadas por mes según el semestre y año seleccionados.

Aclaración: tanto en la vista general como en el detalle, si el recorrido no presenta ningún viaje en el periodo solicitado, la cantidad de cabinas disponibles será 0, por lo que la cantidad de cabinas libres también será 0 y ese es el valor que mostrará.

Cruceros con mayor cantidad de días fuera de servicio

En este utilizo solo las tablas crucero y la tabla crucero_fuera_servicio(en la cual se almacenan los intervalos en los que el crucero ha sido dado de baja), en el WHERE solo igualo que ambas tablas posean el mismo cruc_id(id del crucero), y las funciones anteriormente descritas para seleccionar el año y el semestre, así como con los otros dos listados estadísticos, en el SELECT, selecciono el cruc_id y la suma de la diferencia entre fecha de baja y la de alta lo cual se logra así SUM(DATEDIFF(DAY, CFV.cfs_fecha_baja, CFV.cfs_fecha_alta)), finalmente, para que la suma se pueda realizar, agrupo por cruc_id, lo ordeno descendentemente por la columna en la que se encuentra la sumatoria y selecciono los 5 más importantes con un TOP.
Siguiendo el mismo procedimiento de las demás funcionalidades se muestran 3 los resultados en tres columnas, siendo la tercera la que posee un botón para ver los detalles del crucero.
Al presionar el botón de detalle me muestra una ventana nueva, y utilizando una query muy similar a la anterior, (al igual que la primer funcionalidad) utilizo DATEPART para separa las fechas de baja por mes, contarlas y agruparlas por mes. Esto se mostrará en un DataGridView con la cantidad de días desagregada por mes.

Aclaración: para este punto en particular decidimos insertar información que no se encontraba en la tabla maestra(haciendo tres inserts sobre la tabla de baja crucero, los cuales se encuentran en el script de migración), con el único propósito de poder mostrar

algún valor en este listado estadístico, caso contrario mostraría solo recorridos con cero días.

Pago Reserva

Para mostrar esta vista, la primera acción que realizo es buscar los medios de pago presentes en la tabla medio_pago(la misma está simplemente compuesta por un id numérico autoincremental y el tipo de pago, débito, efectivo, etc, para de esta manera poder en un futuro dado poder expandir el sistema y agregar mas metodos de pago) y esta información la cargo en el ComboBox de medio de pago.
El usuario/ cliente debe ingresar el código de reserva que le fue entregado en la funcionalidad de compra/reserva de pasaje y presionar Buscar en este momento se procede a la validación del mismo, la cual consiste en una validación de tipo de dato(si es entero positivo o otra cosa) y una validación del código de reserva(para efectuar esta evaluación se verifica que la reserva no se encuentre en la tabla de cancelacion_reserva* y que la misma no haya sido pagada ya(validado por un booleano en la misma tabla)).
Luego de validada la reserva, se muestran los errores si los hubiese y si no, los datos correspondientes a la misma, entre los cuales se puede hallar, todos los datos del cliente que creó la reserva y el monto total de la misma(montado hallado a través del precio del recorrido y al que se le aplica el recargo de la/s cabina/s seleccionada/s).
Si el usuario selecciona el medio de pago efectivo, las opciones de cuotas , nro de tarjeta, CVV, etc se bloquean, en cambio si se selecciona el débito, se deben completar los datos necesarios para pagar con tarjeta. Luego de esto se presiona Confirmar, al presionar este botón se validan los datos correspondientes y se persiste un nuevo pasaje, una vez persistido se retorna su id y con el id del pasaje obtenido se persiste la relación cabina_pasaje, utilizando las cabinas seleccionadas para la reserva y el id del pasaje recién agregado.
Una vez persistidas todas las relacion, se realiza un UPDATE sobre la tabla reserva, para asignarle al campo rese_pagada el booleano true. Finalizado todo esto se le muestra una ventana al usuario con un mensaje de éxito e indicándole el código del pasaje recién persistido y se limpian todos los datos de la pantalla para poder continuar pagando más reservas.

*La tabla cancelacion_reserva está compuesta por su PK, la cual es en realidad la FK que apunta a la reserva(ya que no podría cancelar dos veces la misma reserva, lo que vuelve innecesario una PK propia) y un campo de descripción, en el que se anota el motivo de la cancelación