ESTRATEGIA

[FRBA - CRUCERO]

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Alumnos: DUARTE, Gastón

GARCIA, Victoria

MARMO, Matias

MELNYK, Nicolás

[Grupo ZAFFA_TEAM]

Índice:

INTRODUCCIÓN:

En este documento especificaremos la estrategia utilizada para desarrollar un sistema para la comercialización de tickets de cruceros, siguiendo con el Diagrama de Entidad Relación que nos permita crear un modelo de datos que organice y normalice los datos de la Tabla Maestra para una correcta migración de estos e implementación de los nuevos requerimientos a incorporar.

ROLES:

La estructura diseñada para los roles está dada por una entidad a la que denominamos “Rol”, con el nombre como clave primaria y su estado. Este atributo define si el rol está habilitado (‘A’) o inhabilitado (‘I’). A cada rol se le asigna una determinada cantidad de funcionalidades, por lo tanto existe la entidad “Funcionalidad x Rol” que relaciona a un rol con una funcionalidad. Esta entidad intermedia tiene como atributos al nombre del rol (PK del rol) y a la PK de la funcionalidad, definido como int para reducir espacio. Las funcionalidades están definidas por el int que los identifica y su descripción. Este diseño permite entender que un rol puede tener varias funcionalidades y una funcionalidad puede estar asignada a muchos roles.

Como existen dos roles en nuestra base de datos, los únicos nombres de rol definidos son ‘Cliente’ y ‘Administrador General’, permitiendo su relación con las entidades “Cliente” y “Administrativo” respectivamente. Sus nombres son claves foráneas del rol, permitiendo que todos los clientes tengan las mismas funcionalidades asignadas (compra y reserva de pasajes), al igual que en el caso de los usuarios administradores, donde todos tienen todas las funcionalidades permitidas.

Ejemplo:

Para ejemplificar esto, se muestra cómo empezaría todo justo luego de la migración.

Vemos que el “Rol” “Cliente” se encuentra habilitado y este a su vez posee “muchas” funcionalidades (2 en este caso) las cuales son “5” y “8”. Si queremos saber el nombre de las funcionalidades que posee cada uno, solamente vamos a la tabla de Funcionalidad y obtenemos que “5” = “CompraReservaPasaje”, etc.

Administrador general:

El Administrador General cuenta con un username (único) y una contraseña encriptada para hacer posible su login al sistema. Esto fue implementado a través de la función “Hashear_Password”, el store procedure “sp_login” y el trigger “Encriptar_Password” (Explicados en otra sección).

Cuando un usuario intenta loguearse con una contraseña incorrecta, no se le permite acceder a sus funcionalidades y se guarda registro de sus repetidos intentos fallidos. De esta manera, cuando el número de intentos fallidos llega a 3, el estado de ese administrador pasa a estar inhabilitado (‘I’).

En nuestro modelo incorporamos dos usuarios de perfil Administrador, uno con username **‘zaffa_team’** y otro con username **‘admin’**, ambos con password **‘w23e’**.

Cliente:

Los clientes por su parte, no cuentan con un login, por lo tanto al ingresar al sistema se les permite directamente comprar o reservar un pasaje, así también como pagar un pasaje que ya fue reservado ingresando el código de reserva emitido.

El cliente no debe registrarse a la hora de acceder a alguna de las funcionalidades anteriores por el simple hecho de que sería tedioso que si solo queremos ver los viajes que ofrece la app pero no comprarlos, nos tengamos que registrar. Es por eso que la hora de registrarse es cuando se selecciona el viaje y la cabina deseada.

RECORRIDOS:

Nuestro modelo cuenta con las entidades de “Puerto”, “Tramo” y “Recorrido_Unico” que permiten identificar a cada recorrido.

La tabla de puertos cuenta con las filas para su ID (int autoincrementado a través de un identity), su nombre y su estado (activo o inactivo).

Un puerto puede estar en muchos tramos, ya sea como lugar de salida o lugar de llegada. Esto quiere decir que un tramo tiene asignado un ID de puerto de salida (clave foránea de puerto) y otro ID de puerto de llegada (también clave foránea).

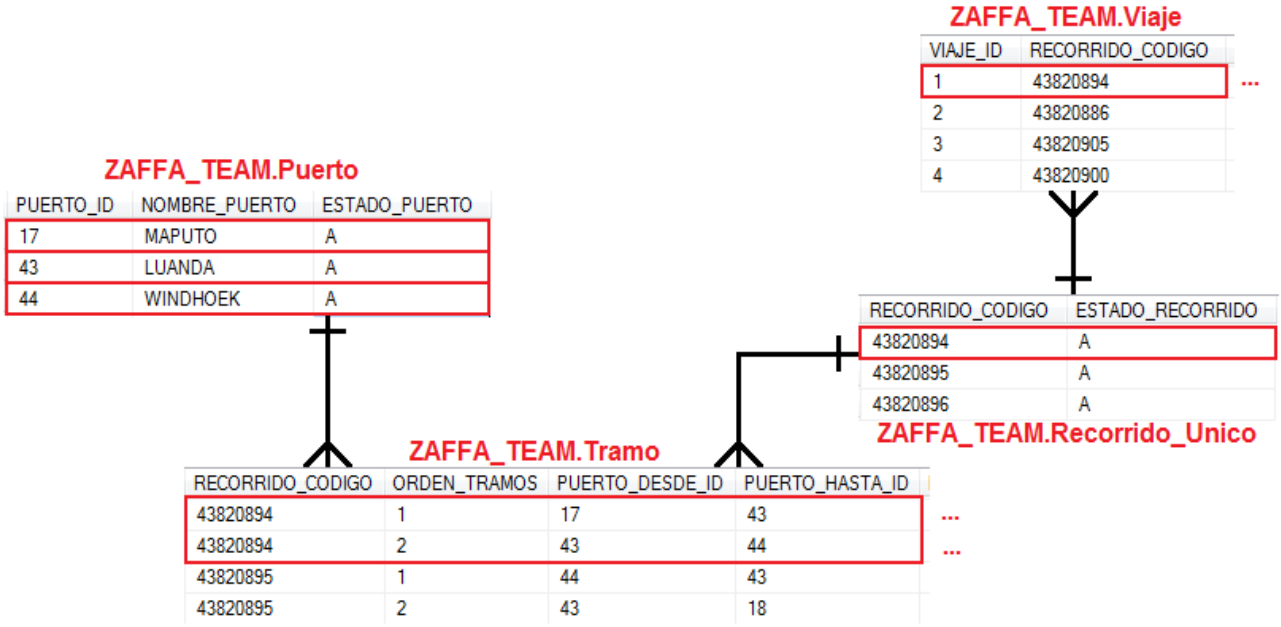
Este tramo también está formado por su precio, el código de recorrido al que pertenece y el orden que tiene en este recorrido. Esto se debe a que un recorrido puede estar formado por uno o más tramos, entonces un recorrido, que puede estar activo (‘A’) o inactivo (‘I’) según su estado, va a estar dado por sus tramos en orden.

Como muchos tramos pueden estar asociados al mismo recorrido, el orden y el recorrido identifican unívocamente a cada tramo (clave primaria compuesta). Distintos tramos pueden tener el mismo puerto de salida (puerto_desde) y mismo puerto de llegada (puerto_hasta), perteneciendo a distintos recorridos y con su respectivo orden.

Se consideró a los tramos que tienen código de recorrido no repetido de orden 1 y aquellos casos que los tramos tienen recorridos repetidos el orden depende de su puerto de salida y su puerto de llegada, donde el tramo que tiene puerto de llegada igual al puerto de salida del tramo con su mismo código de recorrido es el 1 y el otro es el 2, teniendo en cuenta que solo hay hasta 2 tramos distintos con código de recorrido repetidos en la Tabla Maestra. Sin embargo, se asumió que todos los recorridos están formados por un solo tramo, aunque el código de recorrido sea el mismo, ya que hay tramos con igual código de recorrido, donde el de orden 1 tiene fecha de llegada posterior a la fecha de salida del tramo de orden 2.

Ejemplo:

Para poner todas estas palabras en algo más gráfico y sencillo, nos parece conveniente mostrar unas Querys a estas tablas y mostrar los resultados obtenidos.



En este caso consultamos con " SELECT * " las 4 tablas que aparecen en la imagen.

Nos vamos a centrar en el viaje cuyo ID es "1". Vemos que este tiene un único "recorrido_codigo" el cual es "4382094". A partir de este código podemos ver que el recorrido se encuentra habilitado (no fue dado de baja lógica). También este código nos permite apreciar que ese recorrido tiene muchos tramos (2 en este caso), el primero de "17" a "43" y el segundo de "43" a "44". Estos números representan los IDs de puerto, pero si queremos saber el nombre exacto del puerto, a partir de esos IDs lo obtenemos de la tabla puerto quedando "17" = "MAPUTO", etc.

Realizando un JOIN de estas tablas para acomodar los datos relevantes en este caso, nos quedaría algo como lo siguiente que muestra lo explicado anteriormente.

VIAJE_ID	RECORRIDO_CODIGO	NOMBRE_PUERTO	NOMBRE_PUERTO
1	43820894	MAPUTO	LUANDA
1	43820894	LUANDA	WINDHOEK
2	43820886	MONROVIA	LUANDA
2	43820886	LUANDA	TRÍPOLI
3	43820905	VICTORIA	LUANDA

CRUCEROS:

Nuestro modelo cuenta con una tabla denominada “Crucero” donde cada uno de los cruceros es identificado unívocamente según su identificador. Además tiene un modelo asignado, el ID de la marca a la que pertenece (clave foránea de la entidad “Marca” que tiene también una descripción del fabricante), el estado en que se encuentra, la cantidad de cabinas, la fecha de su último cambio de estado y el motivo de estar dado de baja en caso de ser necesario.

El estado del crucero está restringido a través de un constraint donde solo puede tomar los valores de ‘Alta’, ‘Baja Definitiva’, ‘Fuera de Servicio’ y ‘Reinicio de Servicio’. En nuestro modelo, asumimos que todos los cruceros comienzan con el estado de ’Alta’, y en caso de quedar fuera de servicio pueden volver a ser utilizados y su estado pasa a ser ‘Reinicio de Servicio’.

Para mantener un registro de los cambios en el estado que tiene un crucero, modelamos una tabla de auditoría que ante cualquier update en el estado del crucero (funcionalidad permitida solo para aquellos usuarios de rol administrativo) el trigger “Auditoria_de_estado_cruceros” guarda el ID del crucero, la fecha en que cambió su estado, el nuevo estado, la fecha anterior en que se hicieron cambios y su estado anterior. También generamos un trigger que ante la dada de baja definitiva o fuera de servicio del crucero se especifica el motivo, de lo contrario este campo permanece vacío.

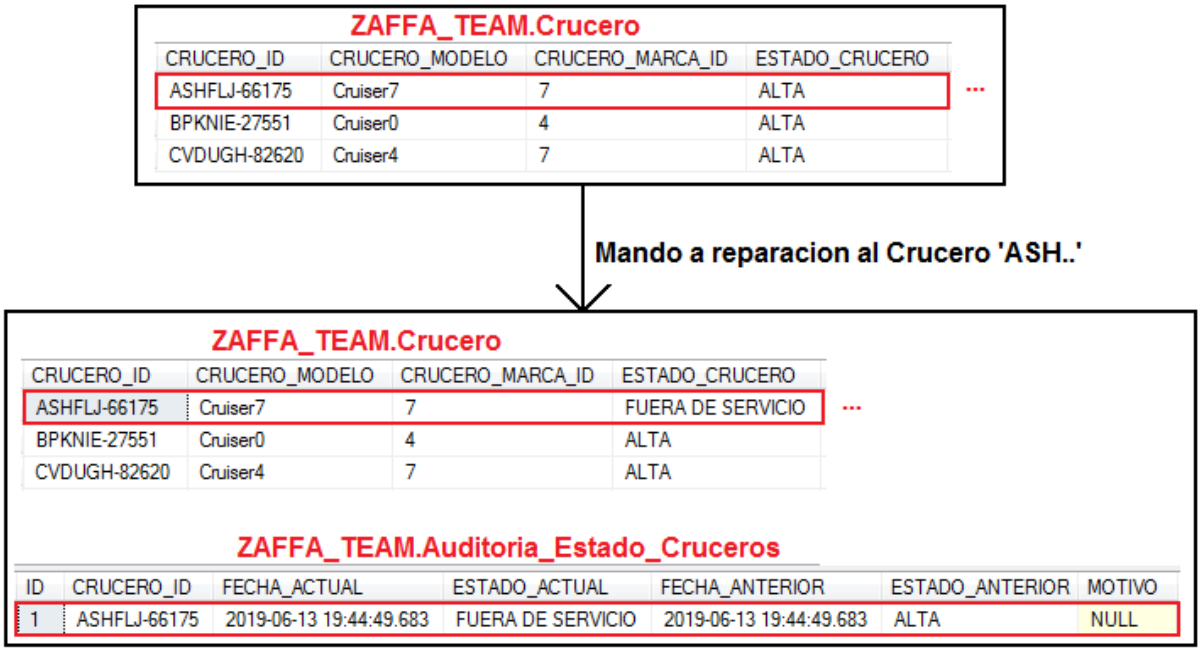
El tipo de servicio de un crucero está dado por sus cabinas. Cada cabina está identificada unívocamente según el crucero al que pertenece, el número y el piso en que se encuentra. También está definido por su tipo, que es clave foránea de la tabla de “Tipo_Cabina”. Esta entidad cuenta con un ID, que cargamos a través de un identity autoincrementado, la descripción de su tipo y el porcentaje de recargo que éste le agrega al precio de venta del pasaje.

Ejemplo 1:

Caso: Quiero mandar a reparar a el Crucero con ID ‘ASH…’

Para resolverlo, le hago un UPDATE al estado del Crucero ese para que cambie a ‘FUERA DE SERVICIO’. Como se ve en las imágenes, efectivamente se cambia el estado del crucero de ‘ALTA’ → ‘FUERA DE SERVICIO’. Esto provoca que se dispare el Trigger que inserta esta modificación en la tabla de Auditorias. Esta tabla es de gran ayuda a la hora de saber porque se cambió de un estado al otro y en que fecha. Un upgrade seria también poder registrar que Administrador hizo este cambio pero ya excede los requerimientos del TP.

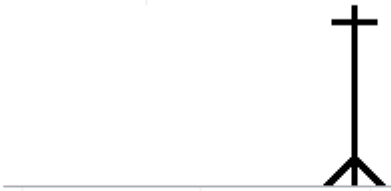
Al hacer un cambio de estado, en la Aplicación nos aparecerá una ventana que nos pide que ingresemos un motivo por el cual se dio este cambio. Esto también nos permite tener un mayor control acerca del estado de los cruceros.



Ejemplo 2:

Si quiero conocer las cabinas del Crucero, puedo acceder mediante el Crucero_ID para ver todas las cabinas que posee. Esta tabla de cabinas tiene una tiple PK que será el CRUCERO_ID, CABINA_NRO, CABINA_PISO.

CRUCERO_ID	CRUCERO_MODELO	CRUCERO_MARCA_ID	ESTADO_CRUCERO	CANTIDAD_CABINAS
ASHFLJ-66175	Cruiser7	7	ALTA	84
BPKNIE-27551	Cruiser0	4	ALTA	44
CVDUGH-82620	Cruiser4	7	ALTA	98



CRUCERO_ID	CABINA_NRO	CABINA_PISO	CABINA_TIPO_ID
ASHFLJ-66175	0	0	2
ASHFLJ-66175	0	1	4
ASHFLJ-66175	1	0	3
ASHFLJ-66175	1	1	2
ASHFLJ-66175	2	0	3
ASHFLJ-66175	2	1	3
ASHFLJ-66175	3	0	5
ASHFLJ-66175	3	1	4

x84

VIAJES:

Un viaje está asociado a un único recorrido y a un único crucero (claves foráneas), pero dado que el mismo recorrido puede estar asignado en distintos viajes realizados por distintos cruceros, la clave primaria es un int identity.

En un nuestro modelo, un viaje también está determinado por su fecha de salida, su fecha de llegada estimada y su fecha de llegada real. De esta forma, solo se puede obtener la fecha de salida del puerto del tramo de primer orden y la fecha de llegada al puerto de último orden, sin tener en cuenta las fechas en que se llega a los puertos intermedios en los casos donde el recorrido de ese viaje está formado por más de un tramo. Sin embargo, si se puede acceder a los puertos intermedios, dejando al usuario bajarse pero no subirse a mitad del recorrido del viaje.

La tabla de viajes cuenta con un índice con el código del recorrido, la fecha de salida, la fecha de llegada y el crucero. Así permitimos a la tabla de pasajes y reservas acceder a sus datos de manera eficiente y más rápida.

RESERVA Y COMPRA DE PASAJES

Las funcionalidades de reserva o compra de pasajes están asignadas tanto para un usuario Administrativo como para un Cliente. De esta forma un administrador puede ingresar sus datos personales, tales como el nombre, apellido, dni, etc. para comprar o reservar un

pasaje y estos son guardados como los de cualquier cliente, a pesar de que ambos son distintos roles.

Explicado esto, el Usuario (Cliente o Administrador General) tiene la posibilidad de reservar o comprar un pasaje, eligiendo la fecha de salida con puerto de inicio, la fecha de llegada al puerto final, el crucero y la cabina.

Una vez seleccionado el pasaje, el usuario puede pagar el pasaje directamente o si desea reservarlo se le devuelve un código de reserva que se mantiene vigente por 3 días. Antes de esos 3 días el usuario puede acceder al sistema, ingresar su código de reserva y pagar el pasaje especificando el medio de pago. En este caso se elimina de la tabla de reservas el pasaje comprado y se agregan a la tabla de pasaje los mismos datos que se encontraban en la reserva, asignándole un nuevo código de pasaje y una fecha de compra. Si pasados esos días el usuario no compro el pasaje la cabina vuelve a estar disponible para su reserva o compra (a través del Store Procedure “sp_borrarReservas”).

Por lo tanto, concluimos que las entidades de “Reserva” y “Pasaje” son similares, con la diferencia de que el pasaje además posee todo el registro de la transacción económica (medio de pago, precio y fecha compra).

TRIGGERS, FUNCIONES Y STORED PROCEDURES RELEVANTES

Hashear Password:

Cuando un administrativo quiere loguearse en la App, este debe ingresar a su “USER” y “PASSWORD”. Automáticamente la App llama al Store Procedure “sp_login” y le pasa los 2 valores ingresados. Este SP toma el valor de la “PASSWORD” y la hashea utilizando la función “Hashear_Password”. Teniendo el USER y la PASSWORD (hasheada) revisa todos los administradores a ver si alguno matchea con lo ingresado. Pueden pasar 3 cosas:

1. Contraseña incorrecta 1 o 2 veces: se le suma 1 a la cantidad de intentos fallidos.
2. Contraseña incorrecta 3 veces: se le cambia el estado al administrador a ‘I’ (inhabilitado) prohibiendo así reingresar a la App.
3. Contraseña correcta: se le da ingreso al sistema y se le setean los intentos fallidos a 0.

Encriptar password:

Este es un trigger que cuando ejecutamos un insert sobre la tabla Administrativo, en lugar de insertarlo, hashea la password y lo inserta.

Auditoria de estado de cruceros:

Descripto brevemente antes, es el encargado de detectar si un crucero tiene un cambio de estado, y en el caso de tenerlo se dispara el trigger que audita guardando del crucero en cuestión su estado anterior, su estado nuevo y las fechas en que ocurrieron las mismas.

Borrar reservas:

Existe un SP que cuando es llamado, borra todas las reservas mayores a 3 días.

Este solo es invocado en 2 partes de la aplicación:

1. Si el usuario es Administrativo: luego de loguearse a la App.
2. Si el usuario es Cliente: cuando este ingresa a la sección de pago de la reserva o de la compra.

Otros:

Por una cuestión de simpleza, nos parecio mas facil crear SP para los distintos INSERTS, UPDATES y DELETES básicos que se realizarán durante el transcurso de la aplicación.

Otra forma podría haber sido realizando los ExecuteNonQuery() desde el C#.

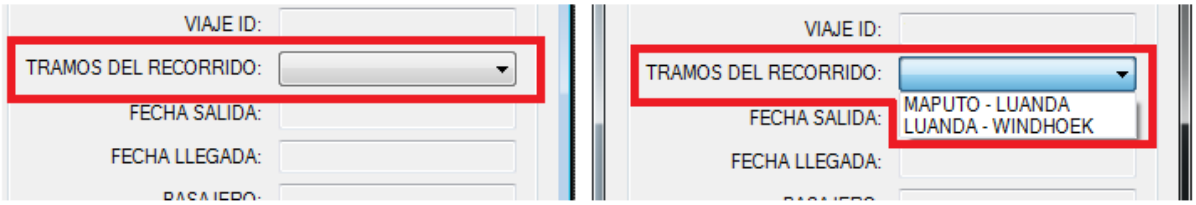
Así, el código de SQL queda con 3 tipos de nombres de SP:

1. sp_guardar…: ejecuta un insert. Ej.: sp_guardarPuerto , sp_guardarRecorrido
2. sp_update..:ejecuta un update.Ej.:sp_updateCantCabinas, Sp_updateNombreRol
3. sp_delete…: ejecuta un delete. Ej.: sp_deleteRol, sp_deleteFuncionalidadxRol

Aspectos de diseño de la aplicación con .NET

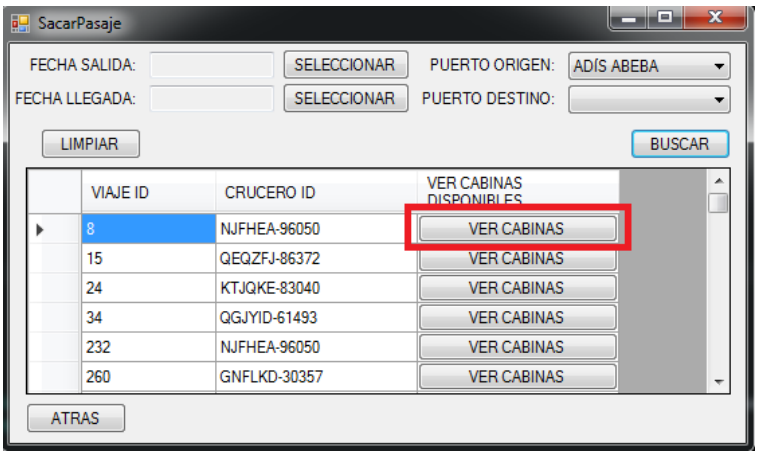
1. Existen varias partes de la aplicación donde es necesario mostrar todos los tramos que posee el viaje. Por ejemplo, a la hora de generar el voucher de compra/reserva, es importante informarle al usuario qué acaba de comprar/reservar.

Ante este requerimiento, decidimos solucionarlo colocando un comboBox que al presionarlo nos muestra todos los tramos ordenados del viaje uno debajo del otro, tal como se muestra a continuación:

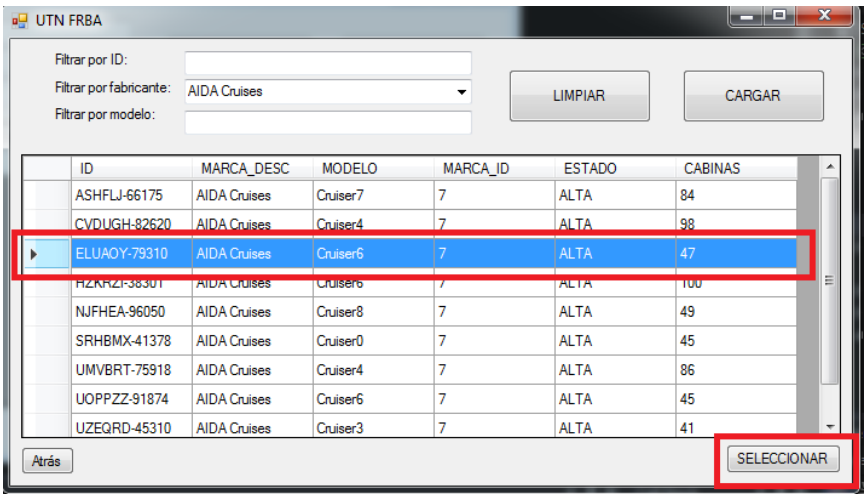


Aclaración: la idea es que el usuario no pueda seleccionar los tramos de ese comboBox, dado que este fue pensado como una estructura informativa. De todas formas, no encontramos forma SW alguna de evitar que puedan ser seleccionados pero no lo vemos como un inconveniente grave, solo como un problema estético.

2. Otra decisión de diseño tomada fue cuando se debe mostrar una grilla con diversos datos que se consultan al SQL. Decidimos tener 2 estilos en la aplicación para ayudar al usuario (a nuestro criterio).
    a. El primero consiste en tener una grilla con los datos mostrados y al final de cada fila un botón que nos direcciona hacia otra pantalla.



    b. El segundo consiste en tener una grilla con los datos mostrados y si queremos seleccionar esa fila para direccionarnos hacia otra pantalla, será necesario clickear la fila deseada y a continuación clickar el botón de continuar.



El hecho de hacer esta distinción de estilos se debe, como mencionamos antes, para simplificarle la operación de la aplicación al usuario.
Se decidió que las tablas las cuales presentan “muchas” columnas tengan la forma de selección B (dado que agregar una columna extra para botones implicaría una pantalla muy ancha y confusa); mientras que las que poseen “pocas” columnas la forma de selección A.