GESTION DE DATOS 2019

- **Cuatrimestre.** 1ro
- **Curso.** K3014.

TRABAJO PRÁCTICO

CRUCEROS FRBA

GRUPO TIRANDO_QUERIES (#24)	
MIEMBRO	**LEGAJO**
LOPEZ SURRA, Rodrigo	152.725-3
CODA, Agustín	155.974-6
ZITO, Alejo	156.626-9
MOSSE, Lautaro	153.902-4

ÍNDICE

MODELADO DE DATOS

DIAGRAMA DE ENTIDAD RELACION

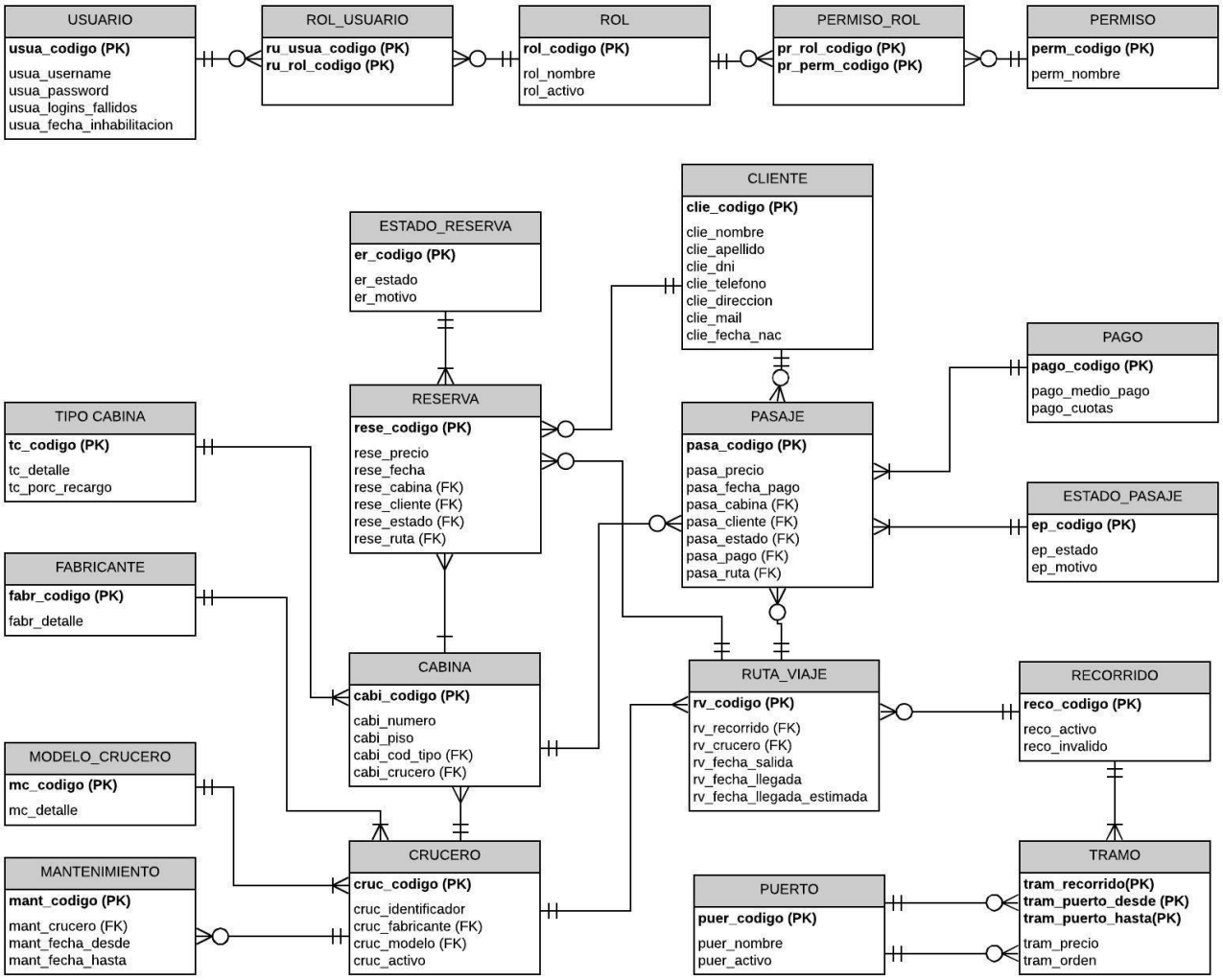

TABLA USUARIO

Almacena información del usuario.

CAMPO	DESCRIPCIÓN
usua_codigo (PK)	Código del usuario
usua_username	Nombre de usuario. Debe ser único.
usua_password	Contraseña del usuario. Almacenada con cifrado SHA256.
usua_logins_fallidos	Cantidad de logins fallidos del usuario
usua_fecha_inhabilitacion	Fecha en que se inhabilita un usuario

ACLARACIONES

- Se definió el set de usuarios:

USERNAME	PASSWORD
admin	w23e
admin2	w23e

TABLA ROL

Almacena información de los roles que permiten al usuario acceder a las distintas funcionalidades de la APP.

CAMPO	DESCRIPCIÓN
rol_codigo (PK)	Código del rol
rol_nombre	Nombre del rol. Debe ser único
rol_activo	Flag que permite identificar la baja lógica de un rol

ACLARACIONES

- Se definieron los roles: Administrativo y Cliente.

TABLA PERMISO

Almacena información de los permisos que permiten al rol acceder a las distintas funcionalidades de la APP.

CAMPO	DESCRIPCIÓN
perm_codig (PK)	Código del permiso
perm_nombre	Nombre del permiso. Debe ser único

ACLARACIONES

- Se definieron los permisos: ABM_ROL, ABM_PUERTO, ABM_RECORRIDO, ABM_CRUCERO, GENERAR_VIAJE, COMPRAR_PASAJE, RESERVA_PASAJE, PAGO_RESERVA, LISTADO_ESTADISTICO.

TABLA ROL_USUARIO

Almacena la relación entre los usuarios y sus roles asociados.

CAMPO	DESCRIPCIÓN
ru_usua_codigo (PK)	Código del usuario
ru_rol_codigo (PK)	Código del rol asociado

ACLARACIONES

- Se definieron las siguientes relaciones:

USUARIO	ROL
admin	Administrativo
admin2	Administrativo

TABLA PERMISO_ROL

Almacena la relación entre los roles y sus permisos asociados.

CAMPO	DESCRIPCIÓN
pr_rol_codigo (PK)	Código del rol
pr_perm_codigo (PK)	Código del permiso asociado

ACLARACIONES

- Se definieron las siguientes relaciones:

ROL	PERMISO
Administrativo	TODOS
Cliente	GESTIONAR_PASAJE, PAGO_RESERVA

TABLA CLIENTE

Almacena información de los clientes.

CAMPO	DESCRIPCIÓN
clie_codigo (PK)	Código del cliente
clie_nombre	Nombre del cliente
clie_apellido	Apellido del cliente
clie_dni	Número de documento del cliente
clie_telefono	Teléfono del cliente
clie_direccion	Dirección del cliente
clie_mail	Mail del cliente
clie_fecha_nac	Fecha de nacimiento del cliente
clie_duplicado	Permite identificar si existe un cliente con DNI duplicado.

ACLARACIONES

- El valor del campo **clie_duplicado** identifica a clientes con DNI duplicados dado que interpretamos el enunciado establece que la aplicación define la univocidad a nivel DNI (regla de negocio). Esto lo vemos, por ejemplo, al momento de la compra de un pasaje: se solicita se completen todos los datos del cliente (si existe en la base de datos) cuando el mismo hace ingreso de su DNI.

TABLA PASAJE

Almacena información de los pasajes comprados.

CAMPO	DESCRIPCIÓN
pasa_codigo (PK)	Código del pasaje
pasa_precio	Precio del pasaje
pasa_fecha_pago	Fecha de compra/pago del pasaje
pasa_cabina (FK)	Código de la cabina al que está asociada
pasa_cliente (FK)	Código del cliente al que está asociado
pasa_estado (FK)	Código del estado al que está asociado
pasa_pago (FK)	Código del pago al que está asociado
pasa_ruta (FK)	Código de la ruta de viaje al que está asociada

TABLA ESTADO_PASAJE

Almacena los distintos estados que un pasaje puede tener.

CAMPO	DESCRIPCIÓN
ep_codigo (PK)	Código del estado del pasaje
ep_estado	Estado del pasaje
ep_motivo	Motivo del estado del pasaje

ACLARACIONES

- Se definieron los siguientes estados:

ESTADO	MOTIVO
Vigente	Pasaje vigente
Desconocido	Sin información
Cancelado	Desperfecto técnico en crucero
Cancelado	Crucero completo VU
Cancelado	Baja de recorrido

- El **estado desconocido** se asocia a los datos migrados, dado que no se tiene la información necesaria para garantizar que hayan sido o no cancelados.

TABLA PAGO

Almacena los distintos tipos de pagos existentes y sus correspondientes cuotas.

CAMPO	DESCRIPCIÓN
pago_codigo (PK)	Código del pago
pago_medio_pago	Medio de pago
pago_cuotas	Cantidad de cuotas para el medio de pago

ACLARACIONES

- Se definió que los pagos se registran colocando una fecha en la tabla de pasaje y referenciando el medio del mismo a esta tabla. La misma limita todos los medios de pago, junto con sus cuotas disponibles. Esto se hizo principalmente para evitar generar datos adicionales sobre entidades poco relevantes al trabajo práctico, y que son brevemente mencionadas en el enunciado.
- Se definieron los siguientes medios de pago:

MEDIO	CUOTAS
Efectivo	0
TD Santander	0
TC Santander	3
TC Ciudad	6
Desconocido	NULL

- El **medio desconocido** se asocia a los datos migrados, dado que no se tiene la información necesaria para garantizar cómo se realizó el pago.

TABLA RESERVA

Almacena información de los pasajes reservados.

CAMPO	DESCRIPCIÓN
rese_codigo (PK)	Código de la reserva
rese_precio	Precio de la reserva
rese_fecha	Fecha de la reserva
rese_cabina (FK)	Código de la cabina al que está asociada
rese_cliente (FK)	Código del cliente al que está asociada
rese_estado (FK)	Código del estado de la reserva al que está asociada
rese_ruta (FK)	Código de la ruta de viaje al que está asociada

TABLA ESTADO_RESERVA

Almacena los distintos estados que una reserva puede tener.

CAMPO	DESCRIPCIÓN
er_codigo (PK)	Código del estado de la reserva
er_estado	Estado de la reserva
er_motivo	Motivo del estado de la reserva

ACLARACIONES

- Se definieron los siguientes estados:

ESTADO	MOTIVO
Vigente	Reserva vigente
Vencida	Supero la cantidad de días permitidos
Pagado	Se realizó el pago
Cancelado	Cliente desiste
Desconocido	Sin información

- El **estado desconocido** se asocia a los datos migrados, dado que no se tiene la información necesaria para garantizar que hayan sido canceladas o vencidas.

TABLA RUTA_VIAJE

Almacena información de las rutas de viajes ofrecidas.

CAMPO	DESCRIPCIÓN
rv_codigo (PK)	Código de la ruta de viaje
rv_recorrido (FK)	Código del recorrido al que está asociado
rv_crucero (FK)	Código del crucero al que está asociado
rv_fecha_salida	Fecha de salida del crucero
rv_fecha_llegada	Fecha de llegada real del crucero
rv_fecha_llegada_estimada	Fecha de llegada estimada del crucero

TABLA RECORRIDO

Almacena los distintos recorridos existentes.

CAMPO	DESCRIPCIÓN
reco_codigo (PK)	Código del recorrido
reco_activo	Flag que permite identificar la baja lógica de un recorrido
reco_invalido	Flag que permite identificar la incerteza de un registro

ACLARACIONES

- Ante la decisión de que cada tramo de la tabla Maestra se considere como un recorrido de un solo tramo (comunicado en el grupo de mail), definimos el flag **reco_invalido** para aquellos recorridos distintos que comparten el recorrido id, dado que no pueden identificarse unívocamente.

TABLA PUERTO

Almacena los distintos puertos existentes.

CAMPO	DESCRIPCIÓN
puer_codigo (PK)	Código de un puerto
puer_nombre	Nombre de un puerto. Debe ser único.
puer_activo	Flag que permite identificar la baja lógica de un puerto

TABLA TRAMO

Almacena los distintos recorridos existentes.

CAMPO	DESCRIPCIÓN
tram_recorrido (PK)	Código del recorrido
tram_puerto_desde (PK)	Código del puerto origen
tram_puerto_hasta (PK)	Código del puerto destino
tram_precio	Precio del tramo
tram_orden	Número de orden del tramo en un recorrido

ACLARACIONES

- El campo **tram_orden** se definió con el objetivo de poder identificar la secuencia de los tramos de un recorrido. Esto es porque en los casos de edición de los recorridos resulta más práctico y más usable (user friendly) de cara al administrador.

TABLA CRUCERO

Almacena información de los cruceros existentes.

CAMPO	DESCRIPCIÓN
cruc_codigo (PK)	Código del crucero
cruc_identificador	Identificador del crucero
cruc_fabricante (FK)	Código del fabricante del crucero al que está asociado
cruc_modelo (FK)	Código del modelo del crucero al que está asociado
cruc_activo	Flag que permite identificar la baja lógica de un crucero

TABLA FABRICANTE

Almacena información de los distintos fabricantes de cruceros existentes.

CAMPO	DESCRIPCIÓN
fabr_codigo (PK)	Código del fabricante
fabr_detalle	Nombre del fabricante

TABLA MODELO_CRUCERO

Almacena información de los distintos modelos de cruceros existentes.

CAMPO	DESCRIPCIÓN
mc_codigo (PK)	Código del modelo
mc_detalle	Nombre del modelo

TABLA MANTENIMIENTO

Almacena información de los mantenimientos de cruceros.

CAMPO	DESCRIPCIÓN
mant_codigo (PK)	Código de mantenimiento del crucero
mant_crucero (FK)	Código del crucero al que está asociado
mant_fecha_desde	Fecha de ingreso en mantenimiento
mant_fecha_hasta	Fecha de egreso de mantenimiento

TABLA CABINA

Almacena información de las cabinas de los cruceros.

CAMPO	DESCRIPCIÓN
cabi_codigo (PK)	Código de la cabina
cabi_numero	Número de la cabina
cabi_piso	Piso de la cabina
cabi_cod_tipo (FK)	Código del tipo de cabina
cabi_crucero (FK)	Código del crucero al que pertenece

TABLA TIPO_CABINA

Almacena los distintos puertos existentes.

CAMPO	DESCRIPCIÓN
tc_codigo (PK)	Código del tipo de una cabina
tc_detalle	Nombre del tipo de cabina
tc_porc_recargo	Porcentaje de recargo del precio

DECISIONES DE DISEÑO E IMPLEMENTACIONES

ARQUITECTURA

La solución se definió en 3 capas: una de presentación, una capa de negocio y, por último, la de acceso a datos. Esta distribución es útil para organizar el código según responsabilidades, y permitiría sustituir fácilmente componentes de ser necesario.

Además, la aplicación desktop se diseñó en base a una arquitectura MVVM de 3 capas: model (domain objects), Views (forms) y ViewModels:

- **MODELS**. Son la representación en POO de las tablas del proyecto. Las clases finalizadas en "DAO" son las encargadas de ejecutar las consultas SQL y retornar las entidades del dominio.
- **VIEWS.** Son los formularios de Windows Forms. Contienen diversos controles como botones o inputs de texto, y se encargan de comunicar los eventos generados por el usuario al operar la UI. Estas clases están encargadas de conectar los diversos componentes visuales con su correspondiente ViewModel. Este proceso se realiza mediante el Binding, donde cada control visual se enlaza a una propiedad del ViewModel y, de esta manera, el objeto está siempre actualizado con la información de la UI.

  En algunos casos, fue necesario realizar Two-Way Binding ya que el ViewModel debía actualizar valores de la vista, como por ejemplo cuándo el usuario ingresa el DNI y, en consecuencia, se debe buscar en la db los datos de Cliente asociados a dicho DNI y autocompletar los campos de texto.
- **VIEW-MODELS.** Son el enlace entre las vistas y los objetos del dominio. Contienen el código correspondiente a la lógica del negocio, como las validaciones que se ejecutan al crear una entidad. Por ejemplo, el cálculo del monto total a medida que el usuario va seleccionando cabinas al comprar un pasaje. Cada formulario, tiene asociado su correspondiente ViewModel de manera de garantizar esta separación de responsabilidades.

En cuanto a la segurización de la aplicación, al ser de escritorio, se definió guardar al usuario logueado en memoria. Dicha clase contiene información del usuario y un listado con todos sus permisos, sobre los que se define cuáles botones de la interfaz visual se muestran y cuáles se ocultan. De esta manera, quedan definidos los accesos a cada sección de la aplicación en base a los permisos del rol del usuario. Si existiera interés en agregar nuevas vistas y segurizarlas, sería cuestión de agregar los permisos a la base de datos y codificar las reglas acordes a estas nuevas vistas.

PROCESO DE LOGIN

El proceso de login se implementa en el stored procedure "SP_AUTENTICAR_USUARIO".

1. Se realiza una validación que el usuario existe y no esté bloqueado. Para este último punto, se verifica si la cantidad de intentos fallidos es mayor o igual a 3.
2. Si la cantidad de intento fallidos es mayor o igual a 3, el usuario quedará bloqueado durante 10 minutos.
3. Se comparan las contraseñas hasheadas.

ASPECTOS VARIOS DE INTERPRETACIÓN DEL ENUNCIADO

- **Conocimiento entre un pasaje y una reserva**. El enunciado no especifica que la reserva y el pasaje necesiten vincularse, dado que no hay ninguna funcionalidad que se valga de esta información. Si fuese necesario, se debiera agregar una FK en la tabla de pasajes, que se complete al momento de pagar una reserva.
- **Edición de cruceros**. Se definió que la edición de cruceros no permite agregar o quitar cabinas dado que resulta ilógico en un escenario real. Es decir, un crucero real no tiene agregados o quitados de cabinas. En tal caso, si una cabina se vuelve no utilizable, se debiera realizar una baja lógica de la misma pero esta situación implica definir una regla de negocio para entender cómo tratar con los pasajes vendidos y la misma no está definida en el enunciado.
- **Reemplazo de cruceros**. Favoreciendo la simplicidad, al enviar un crucero a mantenimiento y optar por su reemplazo, se listan todos los cruceros con suficientes cabinas disponibles para el reemplazo, sin importar los tipos de cabina. Es decir, al reemplazar el crucero pueden reasignarse cabinas con diferente tipo al contratado. Consideramos que el enunciado no explicita una estrategia de reemplazo en especial y que, al tratarse de un caso atípico, el negocio podría permitir dicha regla. Se suma a esto la dificultad para obtener la totalidad de las cabinas por crucero, ya que no contamos con dicha información de a partir de la migración (solamente tenemos las cabinas de aquellos pasajes vendidos que figuraban en la tabla maestra).
- **Mantenimiento**. Favoreciendo la simplicidad, definimos que el mantenimiento de un crucero comienza el mismo día en que este es cargado. Es decir, no se pueden realizar mantenimiento programados siendo que el enunciado no lo menciona. Además, definimos como obligatoria la fecha de egreso de mantenimiento, dado que resulta fundamental a nivel negocio para poder programar futuros viajes.
- **Pago de una reserva**. Favoreciendo la simplicidad, definimos que las reservas se deben pagar en forma individual a pesar de que se pueden realizar varias reservas en una única operación, dado que el enunciado no menciona explícitamente que se requiera de otro modo.
- **Rutas de viaje inválidas**. Al momento de realizar la compra de un pasaje, se listan las rutas de viajes disponibles para los filtros aplicados por el usuario. Se valida que no contengan un recorrido inválido y que el crucero esté en vigencia. Decidimos no poblar la tabla de rutas de viaje con un flag de inválido, dado que depende de información presente en otras tablas y sería redundante.

ASPECTOS GENERALES

- En todos los casos, realizamos validaciones para restringir los datos ingresados por el cliente. Por ejemplo, omitiendo espacios a izquierda y derecha, realizando las comparaciones siempre en mayúsculas, entre otras.
- Cuando en el modelado de datos definimos en las descripciones de algunos campos “debe ser único” esto es una decisión asumida por el grupo dado que, si bien el enunciado no lo menciona explícitamente, entendemos favorece a la coherencia de los datos.
- Para las vistas de los ABMs implementamos una partial class en la que definimos una sola estructura para todos los ABMs y mediante constructor de cada formulario en particular utiliza o no ciertas propiedades del formulario que hereda (Form_Base_Index).

PANTALLAS DE LA APLICACION

INGRESO AL SISTEMA

VISTA DEL ADMINISTRADOR

VISTA DEL CLIENTE