U N I V E R S I D A D T E C N O L Ó G I C A N A C I O N A L
F . R . B . A .
I N G E N I E R I A E N S I S T E M A S D E I N F O R M A C I O N

G E S T I O N D E D A T O S

ESTRATEGIA DE TRABAJO

FRBA HOTEL

FECHA: 11/11/2014

VERSIÓN: 1.2

C O M P U M U N D O _ H I P E R _ M E G A _ R E D

NOMBRE DEL INTEGRANTE	LEGAJO	E-MAIL
GOLOB, Víctor	137.977-0	victorgolob@gmail.com
GEREZ, Claudio	142.558-4	claudiomgerez@gmail.com
FERRARO, Tomás Ignacio	146.518-1	tetferraro@gmail.com
LEDESMA, Nicolás [1]	147.062-0	n.e.ledesma@hotmail.com

[1] Este integrante del grupo fue dado de baja, debido a inactividad y a no responder a los diferentes llamados de participación.

BASE DE DATOS

CREACIÓN DEL NUEVO MODELO DE DATOS Y MIGRACIÓN DE LOS DATOS EXISTENTES

DIAGRAMAS

Partiendo de la **Tabla Maestra** y la documentación del enunciado, se realizó un **modelo desnormalizado** que cumpla las reglas de negocio actuales. Con dicho modelo, se normalizó el mismo hasta **Forma Normal 3** y luego se fueron tomando ciertos recaudos de duplicidad o simplificación de la información. Como ejemplo, uno de ellos fue el utilizar una tabla para los ítems de la factura; que aunque sean consumibles, es conveniente tenerlos separados en una tabla particular como resguardo de información y para evitar cálculos al momento de realizar consultas.

El Diagrama de Entidad-Relación (DER) que representa al modelo, se encuentra adjunto a este documento bajo el nombre de "**DER.png**".

Tras la primera entrega del documento, se lo ha revisado, la relación entre los ítems de las facturas y los consumibles de cada estadía fue alterada de acuerdo a las correcciones realizadas.

MIGRACIÓN DE DATOS

Una vez realizado el nuevo modelo, se analizaron los datos del modelo anterior para poder realizar la migración de los mismos al nuevo sistema. Debido a ciertas **incompatibilidades** entre ambos modelos, se encontraron distintos problemas. A continuación se plantearán los mismos y las **soluciones** tomadas para realizar efectivamente la migración de los datos.

1. **Reservas y Facturas Inválidas**: Se encontraron varias entradas con valores erróneos en las reservas, así como también se encontraron varias entradas de facturas cuyos totales se encontraban mal calculados. Para evitar problemas de estabilidad en el sistema, se los delegó al **sistema de tablas para gestionar las entradas inválidas** y así evitar la pérdida de los datos migrados.

2. **Formato de Emails erróneo**: Todos los emails se encontraban con formatos inválidos para el estándar de la RFC 6531, **conteniendo mayúsculas, espacios y caracteres especiales inválidos** como: é, á, í, ú, ó, ü, etc. Para evitar problemas con el aplicativo, ya que utiliza la clase MailAddress, se alteran los emails con un **trigger** luego de cada insert o update de la siguiente manera:
   - Las mayúsculas se pasan a minúsculas.
   - Los espacios designan el inicio de la cadena.
   - Las vocales con tildes y/o diéresis se colocan en su forma normal.

3. **Datos Faltantes**: Debido a nuevas capacidades del modelo, fue necesario agregar **datos por defecto durante la migración**, como es el usuario que realizo las reservas, cancelaciones, estadías y facturas, como el "admin"; locaciones de los clientes (como son todos argentinos se supuso que provenían de la Ciudad Autónoma de Buenos Aires), DNIs para tipos de documentos, las fechas de creaciones de los hoteles, etc.

4. **Nacionalidades**: Para los Clientes, así como para la nacionalidad, se utiliza el **mismo campo** para definir el **país** de domicilio, a fin de homogeneizar. Además, todos los datos de los campos de nacionalidad, **se cambiaron a forma de país**, ya que el aplicativo posee una lista de todos los países existentes y admisibles.

DECISIONES DE DISEÑO

Junto con la creación del nuevo modelo se tomaron en cuenta ciertas decisiones a la hora de realizar el diseño del sistema. Las mismas fueron:

1. **Inmutabilidad de la Fecha de Creación de un Hotel**: La fecha de creación de los hoteles **no son modificables una vez creadas** debido a que no tiene ningún sentido, más que para subsanar un error. En tal caso, es recomendable realizar la creación del hotel nuevamente y eliminar el erróneo de la Base de Datos.

2. **Validación del Primer Log-in**: La validación del primer Log-in de un Usuario se realiza dentro de la base de datos en vez de la aplicación. Esto permite que sea la Base de Datos la cual decida si modifica o no el password del usuario, ya que por requerimiento "**el password es modificable una vez que el usuario se encuentre logueado**".

3. **Intentos Fallidos de Log-in**: Los intentos fallidos de loggeo se registraran dentro del modelo a fin de realizar una **persistencia** de los mismos, como realizaría un sistema bancario. Esto genera que los mismos no sean volátiles y permite una mayor seguridad.

4. **Interacción con la Aplicación**: Para realizar una capa de abstracción entre nuestro modelo de datos y la aplicación, se crearon varios **Stored Procedures** que serán conocidos por la aplicación **como la interfaz de la Base de Datos.** Esta abstracción **reduce el acoplamiento** entre la aplicación y el modelo de datos, y permite que la aplicación solo conozca qué es lo que necesita y como obtenerlo, sin depender de los nombres de las tablas, sus atributos y relaciones.

5. **Hoteles y Roles**: A fin de simplificar el loggeo y la identificación en el sistema, los **roles** de un usuario son **indistintos** a sus **hoteles**, ya que no tendría sentido práctico aplicable que, por ejemplo, un usuario "Recepcionista" sea "Administrador" en otro Hotel.

TABLAS

Se trató de evitar la creación de tablas de datos Referenciales, debido a que el modelo de datos no posee relevancia en la utilización de los datos, por lo que para el mismo no generan ningún interés. Esto permite que el programador de la Aplicación se encargue de darle relevancia a los datos como el Tipo de Documento, País y Tipo de Cancelación y modifique de acuerdo a la modificación de la Aplicación.

A continuación se detallara el diseño de las tablas por Funcionalidad:

<u>Gestión de Usuarios:</u>

- **Usuarios**: Contiene los usuarios del sistema.
- **Roles**: Contiene los distintos tipos de roles.
- **Roles_X_Usuario**: Relaciona un usuario con uno o más roles.
- **Funcionalidades**: Contiene las funcionalidades del sistema.
- **Funcionalidades_X_Rol**: Relaciona las funcionalidades con los roles.

- **Hoteles_X_Usuarios**: Relaciona los usuarios con los hoteles.
- **Huespedes**: Contiene los huéspedes ingresados en el sistema.

Gestión de Hoteles:

- **Hoteles**: Contiene los hoteles del sistema.
- **Inhabilitaciones**: Contiene las inhabilitaciones de los hoteles.
- **Regímenes**: Contiene los diferentes regímenes de hoteles del sistema.
- **Regímenes_X_Hotel**: Relaciona los hoteles con los regímenes.
- **Habitaciones**: Contiene las habitaciones de los hoteles.
- **Tipo_Habitacion**: Contiene los datos del tipo de habitación.

Gestión de Reservas:

- **Reservas**: Contiene las Reservas del sistema.
- **Detalles_Reserva**: Contiene los detalles de las habitaciones reservadas.
- **Cancelaciones_Reserva**: Contiene las Cancelaciones de las reservas.
- **Estadía**: Contiene la estadía registrada por reserva.
- **Estados_Reserva**: Contiene los diferentes estados por los que pasa una reserva.
- **Consumibles**: Contiene los consumibles registrados en el sistema.
- **Consumibles_X_Estadia**: Relaciona la estadía con los consumibles registrados.

Gestión de Facturas:

- **Facturas**: Contiene las facturas realizadas.
- **Items_Factura**: Contiene los ítems de las facturas realizadas.

Gestión de Datos Erróneos:

- **ReservasInvalidas**: Contiene las Reservas erróneas de la migración.
- **Consumibles_X_Estadia_Invalida**: Relaciona la estadía con los consumibles registrados erróneos de la migración.
- **Detalles_Reserva_Invalida**: Contiene los detalles de las habitaciones reservadas erróneas de la migración.
- **EstadíaInvalida**: Contiene la estadía registrada por reserva erróneas de la migración.
- **FacturasInvalida**: Contiene las facturas realizadas erróneas de la migración.
- **Items_FacturaInvalida**: Contiene los ítems de las facturas realizadas erróneas de la migración.

STORED PROCEDURES

Se crearon varios procedures a fin de servir como una interfaz del modelo de datos ante la Aplicación. Debido a esto, por cada tabla principal, se realizaron 4 **procedures básicos: get, insert, delete y update**. Todos poseen las siguientes condiciones:

- Si se ingresan campos como **-1** para los numéricos y **''** para los varchar() se los considera como "**elementos neutros**" o nulos.

- Los **procedures delete** son lógicos (pone en 0 su CampoBaja) en su mayoría, a excepción de los hoteles, que genera una inhabilitación, y de otros que son bajas físicas (usualmente correspondientes a las Tablas que realizan la Relación de M:M).

- Los **procedures de insert y update**, en campos no PK o no necesarios, no toman en cuenta los campos ingresados como "elementos neutros".

- Los **procedures de get** devuelven todos los campos de la tabla si la PK ingresada es un "elemento neutro".

Además, el sistema posee otros tipos de procedures como:

SP Facturar: Genera una factura para una reserva.

```
@codReserva numeric(18),
@tipoPago   varchar(50),
@codTarjetaCredito   varchar(19) (Se manda el "elemento neutro" en
caso de ser pago en efectivo")
```

SP CancelarReserva: Genera una cancelación a una reserva.

```
@codReserva     numeric,
@motivo         varchar(255),
@fecha          datetime,
@usr         varchar(50),
@estado         int
```

SP getHotelesMayorCancelaciones: Obtiene los 5 Hoteles con Mayor Cancelaciones

```
@opcion  numeric(1),
@anio  numeric(4)
```

*Donde Opción es:

1. Enero a Marzo
2. Abril a Junio
3. Julio a Septiembre
4. Octubre a Diciembre
5. Todo el Año

SP getHotelesMayorConsumiblesFacturados: Obtiene los 5 Hoteles con Mayor cantidad de Consumibles Facturados

```
@opcion  numeric(1),
@anio  numeric(4)
```

SP getHotelesMayorDiasInhabilitado: Obtiene los 5 Hoteles con Mayor Tiempo de Inhabilitación acumulado.

```
@opcion  numeric(1),
@anio  numeric(4)
```

SP getHabitacionesMayorCantOcupadas: Obtiene los 5 Hoteles con Mayor Cantidad de Ocupaciones

```
@opcion  numeric(1),
@anio  numeric(4)
```

SP getMejorCliente: Obtiene los 5 Mejores Huespedes

```
@opcion  numeric(1),
@anio  numeric(4)
```

SP habitacionesDisponibles: Obtiene las habitaciones que se pueden reservar

```
@codHotel   numeric(8),
@tipoHab numeric(18),
@fechaDesde datetime,
@fechaHasta datetime
```

SP hotelEstaVacio: Determina si un Hotel se encuentra vacío en un periodo de tiempo para poder ser inhabilitado

```
@hotel   int,
@inicio datetime,
@fin  datetime
```

FUNCTIONS

Para evitar la duplicación de código debido a poseer tanta cantidad de procedures, se crearon las siguientes funciones:

```
[habitacionesReservadas](@codHotel numeric(8), @fechaDesde datetime,
@fechaHasta datetime) RETURNS @T_HABITACIONESRESERVADAS
TABLE(habitacion numeric(4))
```

Devuelve las habitaciones que se encuentran reservadas de un hotel para un período de tiempo.

```
[corrigeMail] (@s varchar(256)) RETURNS varchar(256)
```

Dado un email, arregla el formato del mismo a una forma estandarizada.

```
[totalConsumibles] (@codReserva  numeric(18)) RETURNS numeric(18,2)
```

Realiza el cálculo del subtotal por consumibles para la factura.

```
[precioRegimen] (@codReserva  numeric(18)) RETURNS numeric(3,2)
```

En función de una reserva, devuelve el precio del régimen del hotel.

```
[recargoEstrella] (@codReserva   numeric(18)) RETURNS numeric(2)
```

Realiza el cálculo del recargo por estrellas del hotel.

```
[porcentualHabitacion] (@codReserva numeric(18)) RETURNS numeric(2)
```

En función de una reserva, devuelve el porcentual del tipo de habitación.

```
[get_ultimoNumeroFactura] () RETURNS numeric(18)
```

Retorna el número de la última factura realizada.

```
[get_ultimoCodigoReserva] () RETURNS numeric(18)
```

Retorna el número de la última reserva realizada.

TRIGGERS

Como se mencionó anteriormente, el sistema solo posee un trigger, siendo el mismo para chequear los emails luego de ser ingresados y ajustar su formato.

APLICACIÓN DESKTOP

INTERFAZ GRÁFICA

DISEÑO GENERAL DE LA APLICACION

La Aplicación Desktop está compuesta por 2 (dos) proyectos, la **interfaz gráfica** (FrbaHotel) y **las Clases de Dominio y Datos** (DOM).

El proyecto DOM está compuesto por una sección de **clases de Dominio**, donde se especifican las clases del modelo de negocios**, clases auxiliares** de Documentos, Dirección, PasswordEncoding (encriptación de contraseñas), Tipo_Habitacion, EstadoReservas e Inhabilitación. Además, posee un módulo de persistencia de datos utilizando el patrón **DAO** (Data Access Objects).

PERSISTENCIA DE DATOS

Como se expuso anteriormente, la persistencia de datos se realiza a partir del patrón de diseño DAO. El mismo consiste en **nuclear la conexión con la Base de Datos** en clases estáticas con el prefijo "DAO" y continuando con el nombre de la Clase de Dominio asociada al objeto, como "DAOUsuario". Estas clases "DAO" heredan su capacidad de conexión y comunicación con la Base de Datos de una **clase abstracta** llamada "**SqlConector**".

Estas clases estáticas poseen métodos para obtener tanto como una **DataTable** como una lista de la clase de dominio; insertar, borrar y actualizar, y algunas clases poseen la capacidad de interactuar con las relaciones M:M. Todos los métodos estáticos hacen referencia a los métodos de comunicación del "SqlConnector" enviando **el nombre del procedure** a ejecutar y pasando por array de objetos los **parámetros** del mismo.

CLASES ESTÁTICAS

Las clases estáticas del sistema se dividen en: las clases de manejo de persistencia de datos; y la clase **Globals**. La segunda mencionada se encarga de tener todas las **funcionalidades comunes** a las ventanas de la **Interfaz Gráfica**, como para realizar el intercambio de ventanas, utilización del **archivo de configuración** para obtener de la cadena con los datos para conectar con la Base de Datos y la fecha del sistema. Además, posee **atributos estáticos globales**, como un array de cadenas de los nombres de los países existentes.

DECISIONES DE DISEÑO

Para esta parte del sistema se consideraron ciertos aspectos no previstos en el enunciado:

1. Cuando un usuario queda **deshabilitado por 3 intentos fallidos** de Login, esta des habilitación es **permanente** hasta que un administrador lo habilite nuevamente.

2. Al ingresar como **invitado** al sistema, se utilizara un pseudo-usuario llamado "GUEST" que será el encargado **representar al agente externo** para realizar las reservas.

3. La **baja de una habitación** es considerada para realizar una reserva; puede haber reservas anteriores que estén ocupando la habitación, pero **no permite reservarla**.

PRUEBAS DE INTEGRACIÓN

ENTRE LA INTERFAZ Y LA BASE DE DATOS

CONCEPTOS GENERALES

Tras **la versión 1.1** de este documento, se integraron al mismo, documentación referida a las **pruebas de integración** entre la aplicación desktop y la base de datos, creada a partir del script.

No se adjuntará información de otro tipo de pruebas debido a la irrelevancia respecto a este documento, ya que **la idea es mostrar las funcionalidades probadas.**

PRUEBAS

Las pruebas se detallarán por funcionalidad a continuación:

Login – *Diseñador: Tomás Ferraro*

- ✓ Iniciar sesión exitoso como administrador y chequear hoteles, roles y funcionalidades.
- ✓ Registrar 3 penalizaciones y ver la baja del usuario.
- ✓ Iniciar sesión como un Invitado y chequear hoteles y funcionalidades.
- ✓ Desloggear un usuario y volver a iniciar sesión.
- ✓ Dar de baja un Rol y verificar que no se encuentre listado con los roles habilitados del usuario.

Tester: Victor Golob.

ABM de Rol – *Diseñador: Tomás Ferraro*

- ✓ Buscar roles filtrando por los filtros de búsqueda a modo de combinatoria.
- ✓ Realizar la alta exitosa de un Rol, chequeando las funcionalidades dadas.
- ✓ Dar de baja un Rol y luego volverlo a habilitar modificándolo.
- ✓ Agregar funcionalidades a un Rol y luego quitarlas. Verificar la persistencia en ambos casos.

Tester: Claudio Gerez.

ABM de Usuarios – *Diseñador: Tomás Ferraro*

- ✓ Buscar usuarios filtrando por los filtros de búsqueda a modo de combinatoria.
- ✓ Realizar la alta exitosa de un Usuario, chequeando los roles y los hoteles dados.
- ✓ Dar de baja un Usuario y luego volverlo a habilitar modificándolo.

Tester: Victor Golob.

ABM de Huéspedes – *Diseñador: Tomás Ferraro*

- ✓ Buscar huéspedes filtrando por los filtros de búsqueda a modo de combinatoria.
- ✓ Realizar la alta exitosa de un Huésped, chequeando los roles y los hoteles dados.
- ✓ Dar de baja un Huésped y luego volverlo a habilitar modificándolo.

Tester: Victor Golob.

ABM de Hotel – *Diseñador: Tomás Ferraro*

- ✓ Buscar un hotel filtrando por los filtros de búsqueda a modo de combinatoria.
- ✓ Realizar la alta exitosa de un Hotel.
- ✓ Inhabilitar a un hotel para un rango de fechas en el que conoce que hay una reserva, para que muestre error.
- ✓ Inhabilitar a un hotel para un rango de fechas en el que el hotel está vacío.
- ✓ Modificar un hotel y verificar la persistencia de datos.

Tester: Victor Golob.

ABM de Habitación – *Diseñador: Tomás Ferraro*

- ✓ Buscar una habitación filtrando por los filtros de búsqueda a modo de combinatoria.
- ✓ Realizar la alta exitosa de una habitación.
- ✓ Dar de baja una habitación y luego volverla a habilitar.

Tester: Claudio Gerez.

ABM de Régimen – *Diseñador: Tomás Ferraro*

- ✓ Buscar un Régimen filtrando por los filtros de búsqueda a modo de combinatoria.
- ✓ Realizar la alta exitosa de un Régimen.

Tester: Victor Golob.

Generar o Modificar una Reserva – *Diseñador: Tomás Ferraro*

- ✓ Verificar que las fechas respeten un rango válido posterior a la fecha de reserva.
- ✓ Verificar que los regímenes posibles a elegir se encuentren habilitados.
- ✓ Buscar disponibilidad de reserva ante un caso donde no alcanzan las habitaciones pedidas.
- ✓ Buscar disponibilidad de reserva ante un caso donde solo haya habitaciones reservadas.
- ✓ Buscar disponibilidad de reserva ante un caso donde se pueda reservar.
- ✓ Verificar que el precio base de la reserva este bien calculado.

- ✓ Verificar que sea posible buscar o registrar a un cliente satisfactoriamente.
- ✓ Verificar que la reserva fue realizada correctamente.
- ✓ Verificar que se puede realizar correctamente la modificación de una reserva y que solamente lo permite si esta se encuentra pendiente o modificada.

Tester: Victor Golob.

Cancelar Reserva – *Diseñador: Tomás Ferraro*

- ✓ Cancelar correctamente una reserva.
- ✓ Verificar los tipos de motivos en función al usuario loggeado.
- ✓ Verificar que no se pueda cancelar una reserva ya efectivizada o una ya cancelada.

Tester: Claudio Gerez.

Registrar Estadía – *Diseñador: Tomás Ferraro*

- ✓ Verificar que solamente se pueda realizar el checkin y checkout en reservas sin confirmar o modificadas.
- ✓ Verificar que si la reserva se encuentra vencida, el sistema genere la cancelación de la misma por No-Show.
- ✓ Verificar que el checkin no se pueda realizar en otra fecha que no sea la de inicio.
- ✓ correctamente una reserva.
- ✓ Verificar los tipos de motivos en función al usuario loggeado.
- ✓ Verificar que no se pueda cancelar una reserva ya efectivizada o una ya cancelada.

Tester: Victor Golob.

Registrar Consumibles – *Diseñador: Tomás Ferraro*

- ✓ Agregar consumible a la estadía correctamente.
- ✓ Verificar que solo se puedan agregar consumibles a reservas con checkin realizado que no se encuentren canceladas.
- ✓ Remover consumibles de una estadía correctamente.

Tester: Claudio Gerez.

Facturar Estadía – *Diseñador: Tomás Ferraro*

- ✓ Verificar que solamente acepte números de estadía válidos (Reservas efectivizadas, ni pendientes, ni modificadas, ni canceladas).
- ✓ Verificar que el precio este correctamente calculado.
- ✓ Realizar la facturación y verificar la persistencia de los datos.

- ✓ Verificar que detalle correctamente los días donde no se utilizó la habitación por haberse retirado antes.

- ✓ Verificar que no permite realizar más de una factura por reserva.

Tester: Claudio Gerez.

Listado Estadístico – *Diseñador: Tomás Ferraro*

- ✓ Verificar que los cinco tipos de criterios sean correctos y su información tenga sentido.

Tester: Claudio Gerez.

REVISIONES

ACTUALIZACIONES ENTRE ENTREGAS

REVISIÓN 1.0

- Revisión Inicial. Entregada en la primera entrega del TP.
- Agregado el Login.
- Agregado el ABM de Cliente.
- Agregado el ABM de Usuario.
- Agregado el ABM de Rol.
- Agregado el ABM de Funcionalidad.

REVISIÓN 1.1

- Agregado el ABM de Hotel.
- Agregado el ABM de Habitación.
- Agregado el ABM de Régimen (Solo Listado y Alta).
- Agregada la capacidad de realizar y modificar Reservas.
- Agregada la capacidad de registrar una estadía.
- Agregada la capacidad de realizar un listado estadístico.
- Agregada la capacidad de registrar consumibles.
- Agregada la capacidad de facturar una reserva.
- Agregada la capacidad de cancelar una reserva.
- Fixes varios en el Login.
- Fixes varios en ABM de Cliente.
- Fixes varios en ABM de Usuario.
- Fixes varios en ABM de Rol.
- Fixes varios en ABM de Funcionalidad.

REVISIÓN 1.2

- Fix en el ABM de Usuario al realizar la alta que no impactaba en la base.
- Fix en el ABM de Usuario debido que la baja siempre habilitada en Modificar.
- Fix en el ABM de Usuario debido a problemas con el filtro por Rol
- Fix en el ABM de Cliente debido a problemas conel filtro por Documento.
- Fix en el ABM de Cliente que no permitía Modificar.
- Fix en el ABM de Cliente al no poder modificar habilitación en Modificar.
- Fix en el ABM de Régimen debido a un typo en el nombre de la Form del Listado.
- Fix en el ABM de Régimen debido a un problema de parse con el precio base de alta de régimen.
- Fix en Reserva, debido a problemas en el SP que determinaba que habitaciones se encontraban libres.
- Fix en Reserva, debido a que era posible modificar una reserva cancelada o efectivizada.
- Fix en Reserva, debido a que no distinguía los regímenes activos de los inactivos y permitía usar estos últimos.
- Fix en CheckIn, debido a que tomaba como fecha de comparación la de la reserva, no la de inicio de la misma.
- Fix en CheckIn, debido a un erróneo chequeo del estado de las estadías. Era posible tomar una reserva efectivizada con checkin como vencida.

- Fix en CheckOut, debido a un erróneo chequeo del estado de las estadías. Era posible tomar una reserva efectivizada con checkin como vencida.
- Cambio en Estadía, anteriormente se utilizaba el CURRENT_TIMESTAMP para registrar las fechas de las estadías, ahora se pasan por parámetro a los SP correspondientes.
- Fix en Registrar Consumibles, había problemas al remover los mismos de una reserva.
- Fix en Registrar Consumibles, no distinguía entre estados de reservas. Era posible registrar consumibles en una reserva cancelada o pendiente.
- Fix en Facturas, existía un error de NULL en el SP totalConsumibles si no había consumibles adheridos a la factura.
- Fix en Listado Estadístico, los trimestres estaban mal dispuestos y no era posible ver el último trimestre o listar por todo el año.
- Fix en Listado Estadístico, al obtener hoteles con mayor cantidad de inhabilitaciones, había un typo en el SP.