

Informe técnico

Segunda iteración de la creación de la base de datos

1. Introducción

El presente documento tiene como objetivo explicar, de manera resumida, el proceso de diseño que se siguió en la segunda versión de la base de datos de la empresa ficticia Free-Tour partiendo desde el diseño conceptual hasta el físico y detallando decisiones que se tomaron en el paso de un diseño a otro. También se explicarán y justificarán decisiones de implementación como los tipos de datos usados en el motor Microsoft SQL Server, los tamaños asociados a cada atributo (cuando corresponda), el uso de ciertas expresiones regulares y el manejo de llaves principales y foráneas.

Debido a la extensión del proyecto este documento no podrá cubrir cada singularidad del diseño, sin embargo, buscará hacer del conocimiento del lector gran parte de este para facilitar su comprensión y que pueda familiarizarse con él. En general el proyecto tiene 3 aspectos principales: el diagrama entidad-relación que representa el modelo conceptual, el diagrama de tablas que representa el diseño lógico y el script de creación de la base de datos escrito con notación propia de del motor de base de datos Microsoft SQL Server.

2. Diseño conceptual

El enfoque en esta iteración fue el de la reducción de la redundancia de datos, la optimización de relaciones entre entidades y la identificación de nuevas necesidades materializadas en historias de usuario. En concreto, se crearon 3 historias de usuario nuevas y se modificó la historia de usuario 9 de la iteración anterior. En la sección siguiente se detallarán cada una de estas historias incluyendo sus títulos, descripciones y criterios de aceptación.

2.1. Historias de usuario

La historia de usuario 9 (véase Figura 2.1.1) añade como especificación que el tipo de localización debe tener una descripción. Esta especificación, aunque en un principio puede parecer innecesaria o incluso redundante (pues es de conocimiento general qué es una ciudad y qué es un municipio), ayuda a usuarios extranjeros a entender la distribución geográfica del país y a proyectar sus viajes de una mejor manera.

Título	Como usuario quiero saber qué tipo de localización estoy visitando, si es una ciudad o un municipio de Colombia, para poder tener una mayor proyección y conocimiento de donde viajaré.
Descripción	Se podrá clasificar a cada localización registrada con su tipo de localización dentro de los dos valores validos (municipio o ciudad), sabiendo que una localización no puede ser los dos tipos al mismo tiempo y que se debe añadir su descripción para brindar más información a turistas.
Criterios de aceptación	<ul style="list-style-type: none"> • Los únicos tipos disponibles son municipio y ciudad. • Cada localización se clasifica en uno y solo un tipo. • La descripción debe ser obligatoria.

Figura 2.1.1. Historia de usuario 9

La historia de usuario 10 establece que cada guía debe ser capaz de registrar los puntos de interés que se visitarán en cada tour realizado por él. Esta implementación le da al guía capacidad de presentar sus tours en la plataforma Free-Tour de mejor manera y ajustarlos de acuerdo con sus necesidades. Por ejemplo, un guía puede estar especializado en ciertos puntos de interés por lo que sus tours pueden ser únicos con recorridos personalizados. La Figura 2.1.2 corresponde a la historia de usuario 10.

Título	Como guía quiero registrar toda la información pertinente de los puntos de interés que se visitarán en cada tour organizado y así ofrecer el catálogo a los turistas de los lugares que conocerán.
Descripción	Los guías serán capaces de utilizar los puntos de interés de los recorridos y los administradores serán capaces de ingresar nuevos sitios de interés con su información más importante, como el estado del punto, los servicios y actividades ofrecidas, el nombre, una descripción, la ubicación geográfica y sus temáticas.
Criterios de aceptación	<ul style="list-style-type: none"> • El estado será uno permitido dentro de los valores ya establecidos. • La descripción puede ser opcional. • Los servicios deben ser establecidos y registrados al punto de interés. • Un tour tiene varios puntos de interés. • Un punto de interés puede ser el inicio de un tour.

Figura 2.1.2 Historia de usuario 10

La historia de usuario 11 especifica las necesidades del administrador de la plataforma a la hora de manejar los usuarios registrados. Un administrador debe de ser capaz de controlar la información de los usuarios eficazmente, ello conlleva que exista una generalización entre los perfiles de los turistas y los guías en una entidad única. En la Figura 2.1.3 se pueden observar estas necesidades reflejadas en la historia de usuario 11.

Título	Como administrador de la base de datos quiero tener una vista menos segmentada de los usuarios que utilizan la plataforma, sin tener en cuenta si este será guía o turista para poder tener más naturalidad a la hora de registrar los datos personales de cada persona.
Descripción	Los administradores de datos serán capaces de poder ingresar nuevos usuarios utilizando sus datos más personales y privados distinguiéndolos o filtrándolos de los datos públicos.
Criterios de aceptación	<ul style="list-style-type: none"> • Se debe registrar los nombres y apellidos de cada usuario. • Su número de identificación debe ser único. • El número de teléfono debe ser único por usuario. • Se podrá añadir la EPS de cada usuario. • Se debe añadir el país de nacimiento de la persona.

Figura 2.1.3 Historia de usuario 11

La historia de usuario 12 complementa lo establecido en la 11; el administrador de la plataforma debe ser capaz también de diferenciar entre guías y turistas. Esto le permite asignar roles y distribuir permisos en Free-Tour, por lo cual es conveniente también hacer una caracterización de perfiles. En la Figura 2.1.4 se organizan estas necesidades.

Hay un aspecto en particular de esta última historia de usuario que es necesario resaltar: los usuarios solo pueden tener un perfil. Seguramente el lector pensará que esta condición es inaudita, pues ninguna plataforma restringiría a un usuario la capacidad de tener más de un perfil. También es posible que se esté preguntando, en el contexto de este proyecto ¿qué significa que un usuario tenga más de un perfil? Para nuestro equipo de trabajo ello significa información repetida en la plataforma; número de documento, teléfono, nombres y apellidos. La plataforma Free-Tour se encarga de almacenar información de turistas y sus perfiles asociados en la plataforma. Perfiles repetidos involucraría información repetida o inconsistente. Por ejemplo, si hay más de un perfil de un mismo guía turístico, ¿cómo sabe el turista cuál perfil debe usar para contactar al turista? De la misma manera, supóngase que existe más de un perfil asociado a un mismo usuario, ¿qué sucede si todos esos perfiles se registran en un mismo tour al mismo tiempo? Este tipo de problemas se producen si la relación entre usuario y perfil se maneja como 1:N.

La relación entre perfiles y usuarios sigue siendo objeto de debate. Personalmente opino que no es óptimo restringir a un usuario la capacidad de tener más de un perfil, por mucho que eso signifique ciertos riesgos en la plataforma (problemas ajenos a la base de datos). Sin embargo, gracias a que una mayoría tiene una postura diferente a la mía, se decidió por mantener la relación tal cual fue descrita.

Título	Como administrador de la base de datos quiero diferenciar si un usuario es un guía o un turista para poder asignar roles y poder diferenciar entre las distintas tareas que estos pueden realizar a través de un perfil.
Descripción	Se podrá guardar datos más generales de los usuarios en su perfil como su foto de perfil, fecha de creación y el email. Además, de poder diferenciar cuando el perfil asociado al usuario es el de un guía o un turista.
Criterios de aceptación	<ul style="list-style-type: none"> • El email de cada perfil será único dado que con dicho email se creará el perfil. • Un usuario puede crear un único perfil. • Un perfil le pertenece a un único usuario. • Cada perfil estará asociado a un guía (en caso de serlo) y cada guía tiene registrados sus propios atributos. • Cada perfil estará asociado a un turista (en caso de serlo) y cada turista tiene registrados sus propios atributos.

Figura 2.1.4. Historia de usuario 12

2.2. Catálogo de entidades, relaciones y especializaciones

De las historias de usuario 9, 10, 11 y 12 se identificaron varias entidades: perfil, usuario, guía y turista. Estas entidades se relacionan entre sí, además de generar nuevas relaciones con otras ya presentes en el diseño anterior. Las relaciones identificadas se describen a continuación:

- **Usuario – Perfil:** la relación es 1:1 con la conjunción “tiene”. Es decir, un usuario tiene un perfil y un perfil tiene un solo usuario asociado.

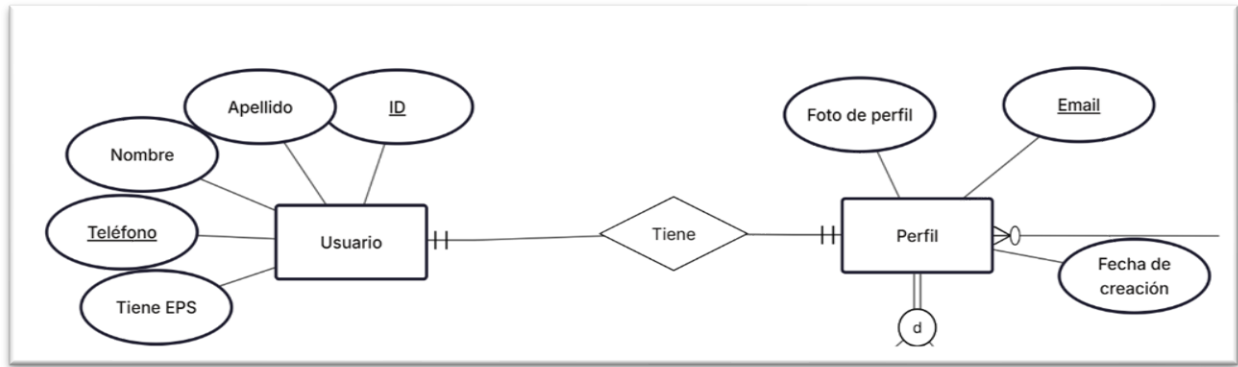


Figura 2.2.1. Relación entre Usuario y Perfil

- **Tour – Puntos de interés (M:N):** la relación es M:N con la conjunción “tiene”. Es decir, un tour tiene varios puntos de interés y un punto de interés está asociado con varios tours.

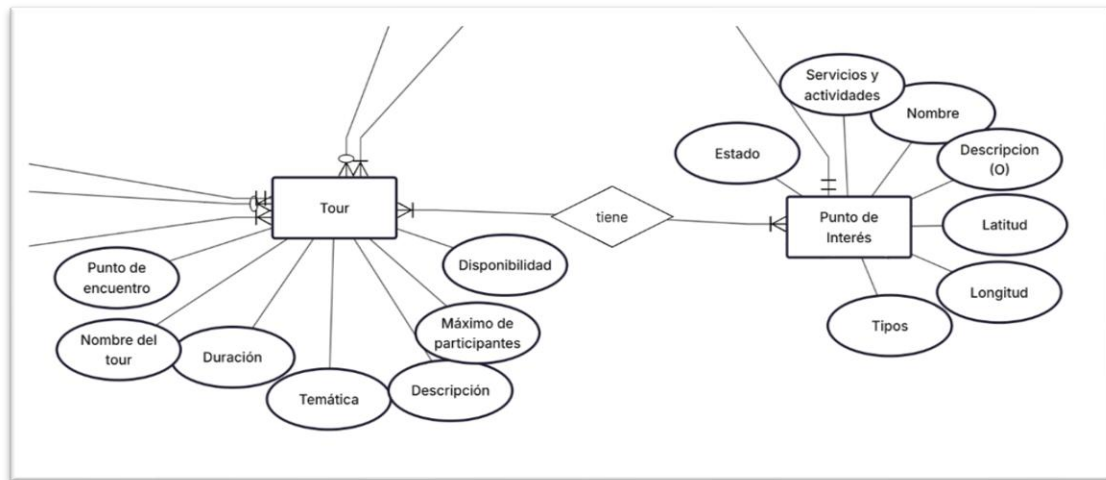


Figura 2.2.2. Relación M:N entre Tour y Punto de interés

- **Tour – Puntos de interés (1:N):** la relación es 1:N con la conjunción “inicia en”. Es decir, un tour inicia en un punto de interés y varios puntos de interés sirven de partida para varios tours.

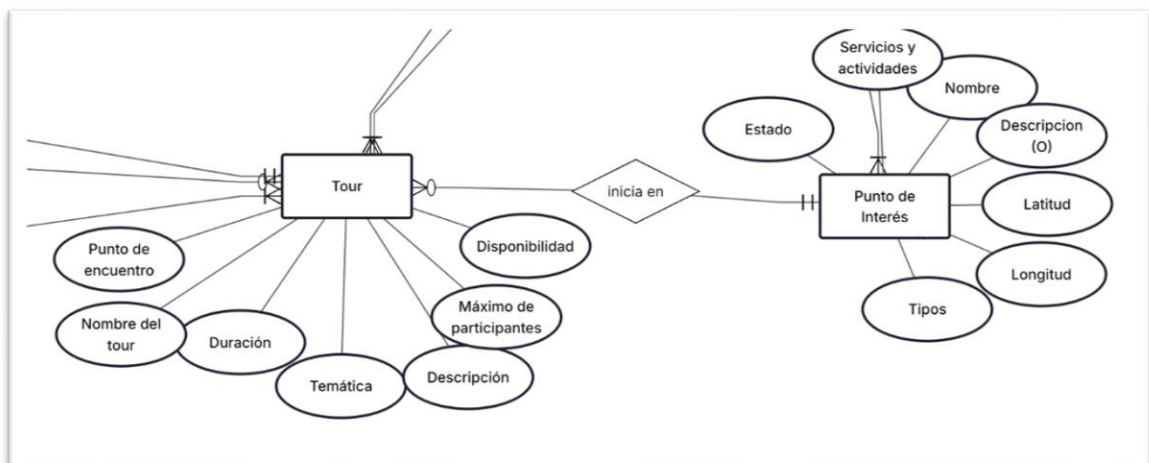


Figura 2.2.3. Relación 1:N entre Tour y Punto de interés

- **Perfil – Idioma:** la relación es N:M con la conjunción “habla”. Es decir, un perfil tiene un usuario que habla varios idiomas y un idioma puede ser hablado por 0 o muchos usuarios que poseen un perfil.

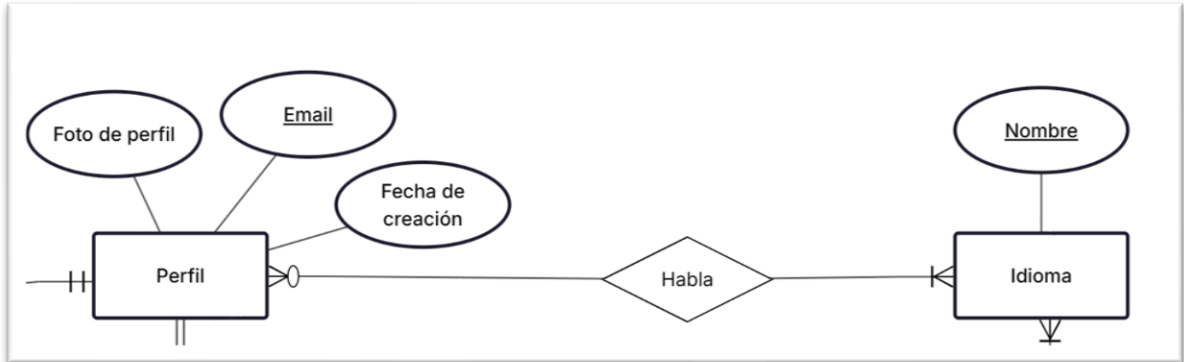


Figura 2.2.4. Relación entre Perfil e Idioma

- **Perfil – Guía, Turista:** se trata de una especialización disjunta total. Cada perfil debe estar relacionado con un solo subtipo y no puede no ser clasificado.

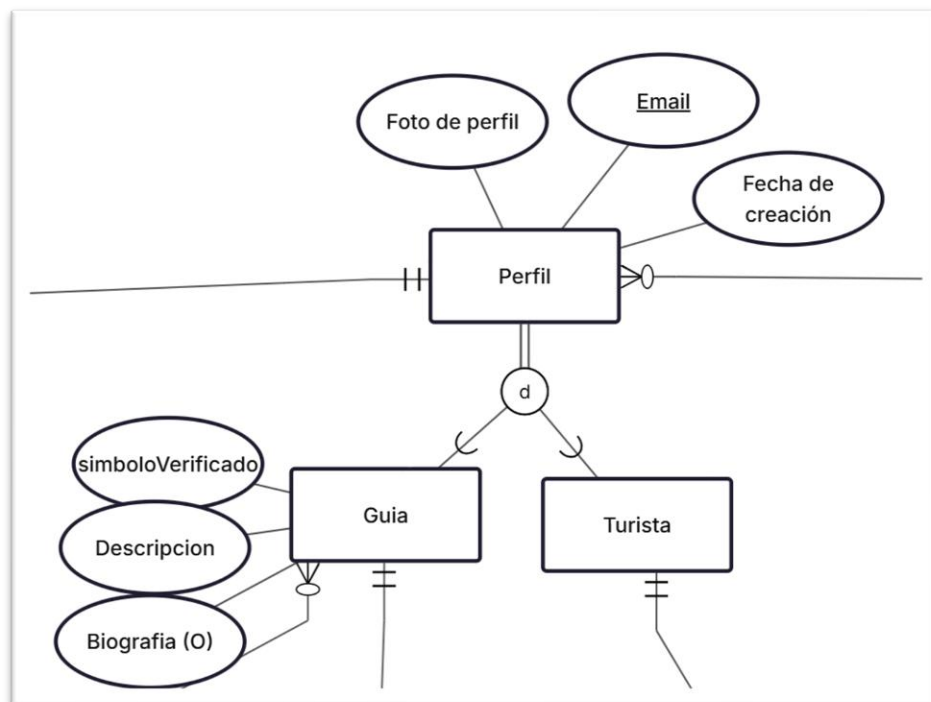


Figura 2.2.5. Relación entre Perfil, Guía y Turista

En general, considerando las entidades de la iteración anterior más las identificadas son:

- Tour
- Guía
- Turista
- Localización
- Tipo de localización
- Reserva

- Idioma
- Usuario
- Perfil
- Punto de interés

La justificación para generalizar guía y turista en una entidad perfil, y generalizar perfil en una entidad usuario es la redundancia de datos. En la iteración previa Guía y Turista compartían un conjunto de atributos en común los cuales son: nombres, apellidos, email, teléfono, foto de perfil, fecha de creación del perfil, etc. Los atributos correspondientes a datos de la persona como tal pueden ser identificados como una entidad por aparte, que en este caso se ha decidido llamar Usuario. Aquellos que corresponden a datos de la plataforma como tal pueden ser asociados a una entidad Perfil. Un perfil contiene información de el usuario que lo maneja, por lo que Perfil es una subentidad de Usuario. Las entidades Guías y Turistas son tipos de perfiles que pueden haber en la plataforma, un perfil corresponde a un turista o a un guía pero nunca a ambas. El razonamiento anterior nos llevó a la conclusión de que el patrón de diseño que se debió seguir para relacionar Guía, Turista y Perfil es el de clasificación total disyunta.

2.3. Matriz de trazabilidad

Todo lo que se ha descrito hasta el momento en el diseño lógico se puede resumir en una matriz de trazabilidad que define los requerimientos que se deben cumplir para que la implementación del diseño en etapas superiores sea efectiva. La matriz en cuestión es la siguiente:

Historia de usuario	Entidades relacionadas	Relaciones
H10: Registrar puntos de interés	Punto de interés, Tour y Guía	Tour – Punto de interés (M:N)
H11: Generalizar usuarios	Usuario	No aplica
H12: Diferenciar guías y turistas	Usuario, Perfil, Guía y Turista	Usuario – Perfil Perfil – Guía Perfil – Turista

Figura 2.3. Matriz de trazabilidad

3. Diseño lógico

En esencia, el diseño lógico consistió en implementar las entidades del diagrama del diseño conceptual en tablas relacionadas utilizando las reglas básicas de conversión. También se optó por añadir un diccionario de datos que luego fue utilizado en el diseño físico para su implementación y se verificó la normalización de las tablas en esta misma etapa.

Las reglas básicas seguidas para pasar del diseño conceptual al lógico fueron: convertir todas las entidades en tablas y los atributos de las entidades en columnas de las tablas; si hay una relación de muchos a muchos, crear una nueva tabla en donde estén las llaves principales de las 2 entidades relacionadas; añadir un identificador a una tabla en caso de que no exista y elegir los tipos de datos correspondientes que permitan representar correctamente el dominio requerido. Estas reglas se siguieron para cada entidad del diseño conceptual por lo que su explicación tabla por tabla se omitirá.

Hay algunas implementaciones que vale la pena mencionar cómo se hicieron. La primera es la generalización entre Guías y Turistas en una sola entidad llamada Perfil. Esta generalización se implementó definiendo a la llave principal de la tabla Perfil como la llave principal y foránea de las tablas Guías y Turistas siguiendo

adecuadamente el patrón de diseño (véase Figura 3.1). Otra es la relación N:M entre Tour y Punto de interés. Esta relación fue implementada creando una nueva tabla llamada PuntosDeInterésDelTour donde se almacenan cada uno de los puntos de interés que están presentes en algún tour (véase Figura 3.2).

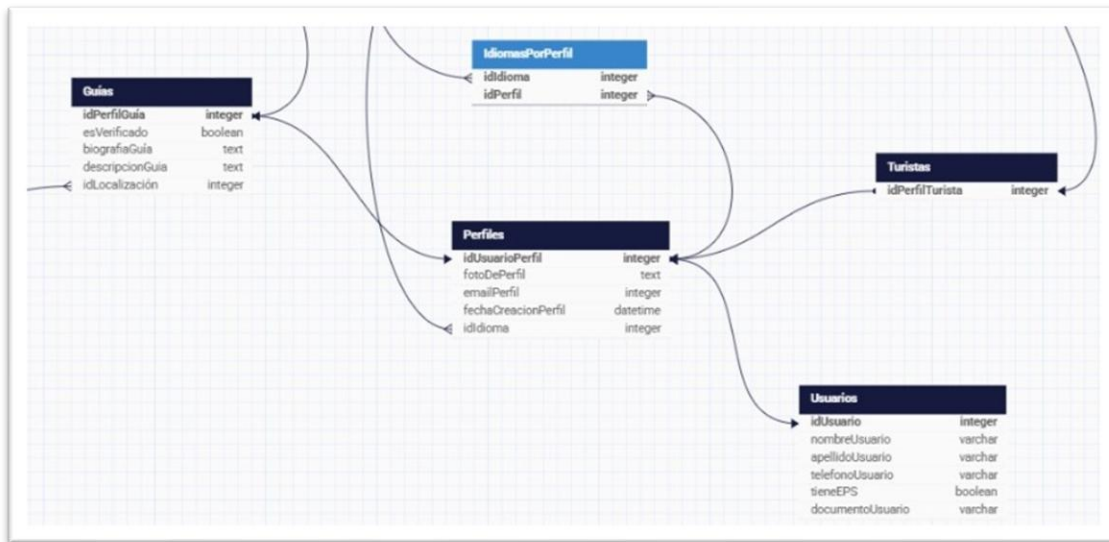


Figura 3.1. Generalización de Guía, Turista y Perfil

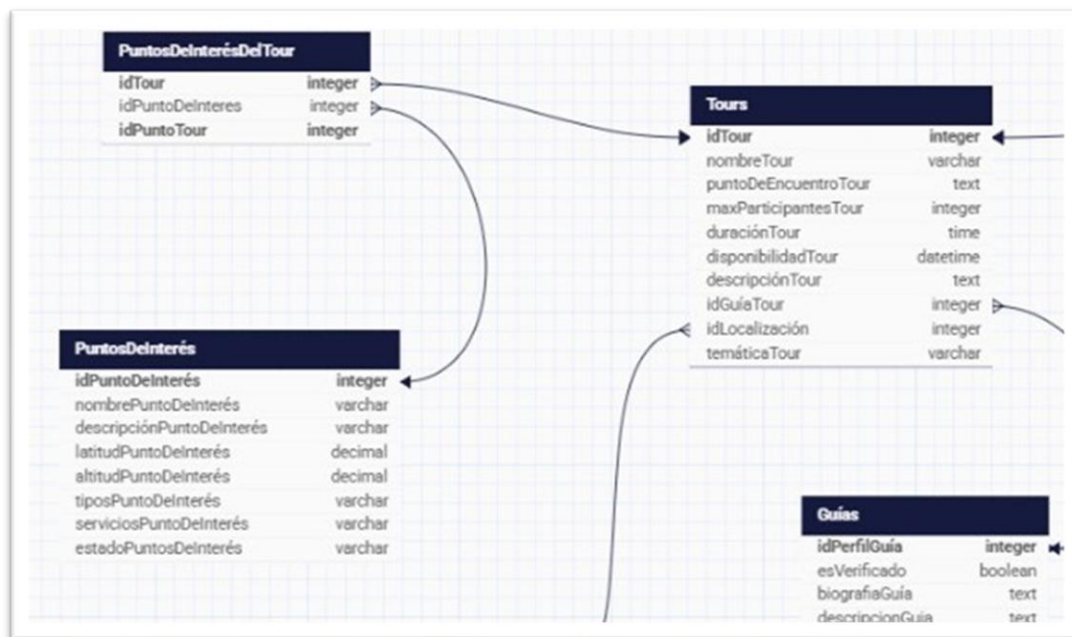


Figura 3.2. Implementación de la relación N:M entre Tour y Punto de interés

3.1. Diccionarios de datos

A continuación, se agrupan el conjunto de diccionarios de datos creados y utilizados en el diseño lógico de la segunda iteración.

Columna	Significado	Valores admitidos	Formato	Reglas aplicables	Ejemplo
---------	-------------	-------------------	---------	-------------------	---------

idTour	Identificador único del tour	Números enteros	Integer	Clave primaria, autoincremental, mayor que 0	1
nombreTour	Nombre comercial del tour	Texto corto	Varchar	Obligatorio, no vacío	“Tour histórico”
temáticaTour		Texto corto	Varchar	Obligatorio, no vacío	“Cultural”
maxParticipantesTour	Cantidad máxima de participantes permitidos en el tour	Números enteros positivos	Integer	Obligatorio, mayor que 0	25
duraciónTour	Tiempo en horas que dura el tour	Números reales positivos	Decimal	Obligatorio, mayor que 0	3.1
disponibilidadTour	Intervalo de tiempo en el cual el tour está disponible	Intervalos de fechas y horas	Varchar	Obligatorio, Formato de fecha y hora válida, 2 fechas, la primera fecha debe ser anterior a la segunda fecha	“9/10/2025 12:08:30 - 10/10/2025 12:08:30 ”
idPuntoDeInterésInicialTour	Identificador del punto de interés en dónde se encontrarán los turistas y el guía.	Número entero	Integer	Obligatorio, debe corresponder a la llave principal de la tabla PuntosDeInterés	1
descripciónTour	Descripción del tour	Texto de longitud considerable	Varchar	Obligatorio, no vacío	“En el tour veremos playas y montañas”
estáActivo	Valor booleano que indica si el tour está activo	true o false	Boolean	Obligatorio, debe ser falso o verdadero	true
idLocalización	Identificador de la localización en donde se realiza el tour	Número entero	Integer	Obligatorio, debe corresponder a la llave primaria de la tabla Localizaciones	1

Figura 3.1.1. Diccionario de la tabla Tours

Columna	Significado	Valores Admitidos	Formato	Reglas Aplicables	Ejemplo
---------	-------------	-------------------	---------	-------------------	---------

idPuntoDeInterés	Identificador del punto de interés	Numérico entero positivo	Integer	Clave primaria, no nulo, incrementa en 1, único	101
nombrePuntoDeInterés	Nombre del punto de interés	Texto corto	Varchar	No nulo	Castillo San Felipe
decripciónPuntoDeInterés	Descripción del lugar	Texto largo	Text	Admite nulos	Imponente fortaleza colonial española
serviciosPuntoDeInterés	Servicios disponibles	Texto largo	Text	No nulo, no vacío	Guías, cafetería, tienda de recuerdos
estadoPuntoDeInterés	Estado actual	“Activo”/“Inactivo”	Varchar	No nulo, no vacío	“Activo”
latitudPuntoDeInterés	Latitud geográfica	Decimal entre -4.2982 y 12.4373	Decimal	No nulo	4.5981
longitudPuntoDeInterés	Longitud geográfica	Decimal entre -78.9909 y -66.8763	Decimal	No nulo	74.0760
tipoPuntoDeInterés	Tipo de sitio	Texto corto	Varchar	No nulo	Castillo

Figura 3.1.2. Diccionario de la tabla PuntosDeInterés

Columna	Significado	Valores Admitidos	Formato	Reglas Aplicables	Ejemplo
idPerfilGuía	Identificador único del guía	Numérico entero positivo	Integer	Clave primaria y foránea que referencia a Perfiles, no nulo, único	301
esVerificado	Indica si el guía está verificado	True/false	Boolean	Por defecto false	True
biografíaGuía	Biografía del guía	Texto largo	Text	Admite nulos	Guía con 10 años de experiencia en tours históricos
idLocalizaciónGuía	Identificador de la localización donde se ubica el guía	Numérico entero existente en la tabla Localizaciones	Integer	No nulo, debe corresponder a la llave primaria de la tabla Localizaciones	1

Figura 3.1.3. Diccionario de la tabla Guías

Columna	Significado	Valores Admitidos	Formato	Reglas Aplicables	Ejemplo
idPerfilTurista	Identificador único del guía	Númerico entero positivo	Integer	Clave primaria y foránea que referencia a Perfiles, no nulo, único	1

Figura 3.1.4. Diccionario de la tabla Turistas

Columna	Significado	Valores Admitidos	Formato	Reglas Aplicables	Ejemplo
idReserva	Identificador único de la reserva	Número entero positivo	Integer	Clave primaria, no nulo, incrementa en 1, único	1
idTurista	Identificador del turista que hizo la reserva	Número entero existente en la tabla Turistas	Integer	Clave foránea a la tabla Turistas, no nulo	1
idTour	Identificador del tour reservado	Número entero existente en la tabla Tours	Integer	Clave foránea a la tabla Tours, no nulo	1
cuposReservados	Cantidad de cupos reservados	Número entero positivo	Integer	No nulo	12
estadoReserva	Estado en el que se encuentra la reserva	“Por confirmar”/ “Aceptada”/ “Finalizada”	Varchar	No nulo. No vacío, cambia de valor	“Aceptada”

Figura 3.1.5. Diccionario de la tabla Reservas

Columna	Significado	Valores Admitidos	Formato	Reglas Aplicables	Ejemplo
idLocalización	Identificador único de la localización	Número entero positivo	Integer	Clave primaria, no nulo, incrementa en 1, único	1
nombreLocalización	Nombre de la localización	Texto corto	Varchar	No nulo, no vacío	Bogotá
longitudLocalización	Longitud geográfica	Decimal entre -78.9909 y -66.8763	Decimal	No nulo	74.8620
latitudLocalización	Latitud geográfica	Decimal entre -4.2982 y 12.4373	Decimal	No nulo	4.5981

idTipoLocalización	Identificador del tipo de la localización	Número entero existente en la tabla TiposDeLocalización	Integer	No nulo, clave foránea a la tabla TiposDeLocalización	1
--------------------	---	---	---------	---	---

Figura 3.1.6. Diccionario de la tabla Localizaciones

Columna	Significado	Valores Admitidos	Formato	Reglas Aplicables	Ejemplo
idTipoDeLocalización	Identificador único del tipo de localización	Número entero positivo	Integer	Clave primaria, no nulo, incrementa en 1, único	1
nombreTipoDeLocalización	Nombre del tipo de localización	Texto corto	Varchar	No nulo, no vacío	Ciudad

Figura 3.1.7. Diccionario de la tabla TiposDeLocalización

Columna	Significado	Valores Admitidos	Formato	Reglas Aplicables	Ejemplo
idPerfilUsuario	Identificador único del perfil	Número entero existente en la tabla Usuarios	Integer	Clave primaria y foránea que referencia a la tabla Usuarios	1
fotoDePerfil	Foto que sirve para representar al perfil	Cadena de texto como URL o ruta	Varchar	Admite nulos	img\imagen.png
emailPerfil	Correo registrado del perfil	Dirección de correo electrónico	Varchar	No nulo, no vacío, único	perfil@correo.com
fechaCreaciónPerfil	Fecha de registro del perfil	Fecha entre la creación de la plataforma y la actualidad	Datetime	No nulo	11/02/2023 12:55

Figura 3.1.8. Diccionario de la tabla Perfiles

Columna	Significado	Valores Admitidos	Formato	Reglas Aplicables	Ejemplo
idUsuario	Identificador único del usuario	Número entero positivo	Integer	Clave primaria, no nulo, único, incrementa en 1	1
nombreUsuario	Nombre del usuario	Cadena de texto	Varchar	No nulo, no vacío	Max

apellidoUsuario	Apellido del usuario	Cadena de texto	Varchar	No nulo, no vacío	Verstappen
documentoUsuario	Documento de identidad del usuario	Secuencia de caracteres	Varchar	No nulo, no vacío, único	545.254.145
teléfonoUsuario	Teléfono registrado del usuario	Número de teléfono válido con su indicativo	Varchar	No nulo, único, no vacío	+57 3145483510
tieneEPS	Valida si el usuario cuenta con EPS	True/false	Boolean	No nulo	False

Figura 3.1.9. Diccionario de la tabla Usuarios

Columna	Significado	Valores Admitidos	Formato	Reglas Aplicables	Ejemplo
idIdioma	Identificador único del idioma	Número entero positivo	Integer	Clave primaria, no nulo, único, incrementa en 1	1
nombreIdioma	Nombre del idioma	Idiomas existentes	Varchar	No nulo, no vacío	Inglés

Figura 3.1.10. Diccionario de la tabla Idiomas

Columna	Significado	Valores Admitidos	Formato	Reglas Aplicables	Ejemplo
idIdioma	Identificador del idioma	Número entero existente en la tabla Idiomas	Integer	Clave primaria y foránea que referencia a la tabla Idiomas	1
idPerfil	Identificador del perfil	Número entero existente en la tabla Perfiles	Integer	Clave primaria y foránea que referencia a la tabla Perfiles	1

Figura 3.1.11. Diccionario de la tabla IdiomasPorPerfil

Columna	Significado	Valores Admitidos	Formato	Reglas Aplicables	Ejemplo
idIdioma	Identificador del idioma	Número entero existente en la tabla Idiomas	Integer	Clave primaria y foránea que referencia a la tabla Idiomas	1
idTour	Identificador del tour	Número entero existente en la tabla Tours	Integer	Clave primaria y foránea que referencia a la tabla Tours	1

Figura 3.1.12. Diccionario de la tabla IdiomasPorTour

Columna	Significado	Valores Admitidos	Formato	Reglas Aplicables	Ejemplo
idPuntoDeInterésTour	Identificador único del punto de interés de un tour	Número entero positivo	Integer	Clave primaria, único, no nulo, incrementa en 1	1
idPuntoDeInterés	Identificador del punto de interés	Número entero existente en la tabla PuntosDeInterés	Integer	Clave foránea que referencia a la tabla PuntosDeInterés	1
idTour	Identificador del tour	Número entero existente en la tabla Tours	Integer	Clave foránea que referencia a la tabla Tours	1

Figura 3.1.13. Diccionario de la tabla PuntosDeInterésPorTour

3.2. Normalización

Procedemos a distinguir en qué forma normal se encuentra cada tabla:

- **Usuarios:** no se encuentra normalizada, ya que la columna teléfonoUsuario no es atómica, es decir, puede contener más de un dato; lo que hace que un mismo usuario se repita tantas veces como cantidad de teléfonos tenga si se obliga la atomicidad manteniendo esta estructura.
- **Perfiles:** no se encuentra normalizada, ya que la columna emailPerfil no es atómica, es decir, puede contener más de un dato; lo que hace que un mismo perfil se repita tantas veces como cantidad de emails tenga si se obliga la atomicidad manteniendo esta estructura.
- **Idiomas:** se encuentra en tercera forma normal, ya que cumple con el principio de atomicidad, además de no contar con dependencias parciales ni transitivas.
- **Guías:** se encuentra en tercera forma normal, ya que las columnas son indivisibles y dependen única y completamente de la clave primaria. Obsérvese que, en dado caso de almacenar en esta misma tabla más datos de la localización del guía, se presentarían dependencias transitivas, lo que no permitiría la 3FN.
- **Turistas:** la tabla está formada únicamente por su clave primaria, por lo que no es posible que haya dependencias parciales ni transitivas, y por definición esta columna es atómica; por lo que podemos entender que esta está en 3FN.
- **Reservas:** no se encuentra normalizada, ya que las reservas pueden cambiar su estado, por lo que se presentarán registros repetidos de las reservas según la cantidad de cambios que tenga en su estado.
- **Tours:** se encuentra en 3FN, ya que cada columna contiene un único valor, además de depender única y completamente de la clave primaria. Véase que la columna puntosDeInterésTour podría presentar conflictos en la normalización al ser posible entenderla como una lista de datos, no obstante, esto se soluciona al agregar la tabla intermedia PuntosDeInterésPorTour, la cual tendrá una relación 1:N con Tours y PuntosDeInterés.

- **PuntosDeInterés:** no está normalizada, ya que la columna serviciosPuntoDeInterés puede almacenar más de un dato por cada punto de interés, lo que haría que este punto se repita tantas veces como servicios tenga si se obliga la atomicidad manteniendo esta estructura.
- **Localizaciones:** todas sus columnas son atómicas y dependen única y completamente de la clave primaria, por lo que está en 3FN.
- **TiposDeLocalización:** todas sus columnas son atómicas y dependen única y completamente de la clave primaria, por lo que está en 3FN.

Nótese que las tablas intermedias que surgen de las relaciones N:M están normalizadas por definición, por lo que no son incluidas en este análisis.

3.3. Tamaños elegidos para ciertos datos

Hay algunos tamaños para ciertos datos que fueron escogidos con base en las necesidades de almacenamiento y del buen manejo de la base de datos. Las decisiones involucradas con la asignación de tamaños se resumen en la siguiente lista:

- **Nombres generales:** para nombres de datos generales (como localizaciones y tipos) se optó por otorgar como espacio máximo 100 caracteres de longitud.
- **Nombres y apellidos de personas:** para nombres y apellidos de personas se optó por otorgar como espacio máximo 50 caracteres de longitud para cada dato. A pesar de que existen nombres que pueden llegar hasta los 1000 caracteres de longitud (https://people-howstuffworks-com.translate.goog/longest-name-in-the-world.htm?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc), son casos tan atípicos que no merecen ser considerados en la base de datos por temas de eficiencia. 50 caracteres son más que suficientes para cubrir la inmensa mayoría de nombres y apellidos alrededor del mundo y, en general, muchas plataformas manejan tamaños similares.
- **Teléfonos:** los números de teléfono escritos con formato pueden constituir cadenas de entre 11 a 18 caracteres de acuerdo con el estándar internacional ITU-T E.164 (<https://www.quora.com/Which-countries-have-the-shortest-and-longest-phone-numbers-country-code-included>). Es por ello que se decidió por almacenar números de teléfono como cadenas con una longitud máxima de 20 caracteres. Esos 2 caracteres adicionales son para manejar posibles casos excepcionales.
- **Descripciones y biografías:** las descripciones y biografías que se almacenen no superarán los 1000 caracteres por decisión de diseño. Consideramos que 1000 caracteres son suficientes para almacenar una descripción de una entidad, sea cual sea en la plataforma.
- **Emails:** el número máximo de caracteres permitido para registrar un email es de 320 por lo que en nuestra base de datos ese será el límite de las cadenas que almacenan emails.
- **Documentos:** los números de documentos tienen a lo mucho 11 dígitos dependiendo del país. En la base de datos se optó por almacenar documentos como una cadena de como máximo 25 caracteres para admitir especificaciones adicionales y formato en el registro. Por ejemplo, cadenas como “C.C. 1082.893.372” pueden ser almacenadas.
- **Coordenadas:** la latitud y la longitud ingresadas a modo de coordenadas son guardadas con 4 dígitos de precisión decimal. Dado que la latitud oscila entre los valores -90 y 90, se almacenan 2 dígitos a la izquierda de la coma decimal. La longitud oscila entre -180 y 180, por lo que se almacenan 3 dígitos a la izquierda de la coma decimal.

4. Diseño físico

El diseño físico consistió en programar el script de creación que convierte las tablas del diseño lógico en tablas implementadas en el DBMS. Durante este proceso hubieron ciertos tipos de datos que fueron seleccionados para satisfacer las necesidades de almacenamiento y a su vez facilitar la mantenibilidad del proyecto. También se implementaron ciertas reglas de acuerdo con las herramientas ofrecidas por SQL Server para este fin, sin embargo, debido a ciertas limitaciones, algunas reglas no pudieron ser estrictamente implementadas.

En esta sección me dedicaré a explicar los aspectos más relevantes del script de creación, cómo se transformaron las columnas de las tablas del diseño lógico en tablas del diseño físico, qué decisiones se tomaron a la hora de aplicar reglas sobre columnas y de qué manera se estructuró el script a fin de promover la legibilidad de este.

4.1. Transformación de datos lógicos en datos físicos

El diseño lógico se creó según tipos de datos generales del estándar SQL; en la práctica, la mayoría de estos tipos de datos varían según el DBMS utilizado. Por ejemplo, en SQL Server no existe el tipo BOOLEAN, en su lugar se utiliza el tipo BIT. A veces el DBMS implementa mejoras con respecto al tipo de dato estándar mediante un nuevo tipo, tal es el caso de TEXT y VARCHAR(MAX) en SQL Server.

Así, dado que el proyecto está implementado en SQL Server se puede resumir que la transición de datos lógicos a físicos se realizó mediante las siguientes consideraciones:

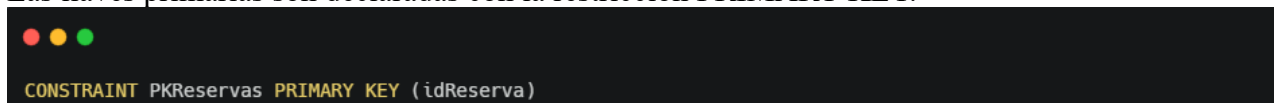
- Si el tipo de dato es TEXT, implementar en el DBMS como VARCHAR(MAX).
- Si el tipo de dato es BOOLEAN, implementar en el DBMS como BIT.
- Si el tipo de dato es DATETIME, implementar en el DBMS como DATETIME2.
- El resto de los tipos se implementan tal cual están en el diseño lógico con los tamaños correspondientes.

4.2. Reglas implementadas

Las reglas especificadas en los diccionarios de datos del diseño lógico se implementaron como CONSTRAINTs en el script de creación. Un detalle que el lector probablemente notará al revisar el script es que casi todas las reglas son CONSTRAINTs en lugar de simplemente declararlas en la definición de cada columna (como las PRIMARY KEYs y las FOREIGN KEYs), la razón de esta particularidad es simplificar la mantenibilidad del código a lo largo del tiempo y asociarle nombres a cada restricción que tiene una columna. De esta manera, cuando una restricción es vulnerada, el DBMS se encargará de identificar con mayor eficacia qué tipo de restricción se vulneró.

Como se explicó previamente, no todas las reglas fueron implementadas tal cual se requería por ciertas limitaciones físicas del lenguaje SQL. Las reglas implementadas son:

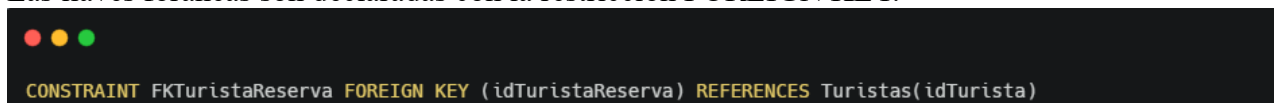
- Las llaves primarias son declaradas con la restricción PRIMARY KEY.



```
CONSTRAINT PKReservas PRIMARY KEY (idReserva)
```

Figura 4.2.1. Script de ejemplo 1

- Las llaves foráneas son declaradas con la restricción FOREIGN KEY.



```
CONSTRAINT FKTuristaReserva FOREIGN KEY (idTuristaReserva) REFERENCES Turistas(idTurista)
```

Figura 4.2.2. Script de ejemplo 2

- Los números de teléfono son validados con la restricción LIKE '+[0-9][0-9] %[0-9]%' OR LIKE '+[0-9][0-9][0-9] %[0-9]{}'. Esta restricción obliga al usuario a ingresar cadenas que empiecen en el símbolo + seguido de 2 o 3 dígitos, luego un espacio de seguido y finalmente una cadena de cualquier cantidad de caracteres que tenga al menos un dígito entre sus caracteres. Como se puede notar, esta restricción no es del todo funcional pues admite cadenas como '+57 a123b', pero es una gran aproximación a lo requerido. La razón por la cual esto se implementó así es debido a la ausencia de características propias de las expresiones regulares que permitirían crear una restricción más apropiada.

```

CONSTRAINT TeléfonoVálido CHECK (teléfono LIKE '+[0-9][0-9] %[0-9]%'
OR teléfono LIKE '+[0-9][0-9][0-9] %[0-9]{}')

```

Figura 4.2.3. Script de ejemplo 3

- Los emails son validados con la restricción LIKE '_%@_%. %{}'. En este caso se fuerza a que el email ingresado sea una cadena de texto que tenga al menos un carácter previo a una @, un carácter posterior a la @ y previo a un símbolo de puntuación, y un carácter posterior al signo de puntuación. Al igual que con la validación de los números de teléfono, esta restricción restringe el dominio de valores permitidos por la columna emails pero aún sigue siendo bastante flexible y poco eficaz gracias a las limitaciones de las expresiones regulares de SQL.

```

CONSTRAINT EmailVálido CHECK (email LIKE '_%@_%. %{}')

```

Figura 4.2.4. Script de ejemplo 4

- Las coordenadas ingresadas en una localización son validadas mediante la restricción latitud BETWEEN -4.2 AND 13.5 AND longitud BETWEEN -79.0 AND -66.0. De acuerdo con los datos registrados en la página <https://www.geodatos.net/coordenadas/colombia>, Colombia se encuentra ubicada en la coordenada (4.570868, -74.297333). Mediante el rango de valores propuestos, las coordenadas ingresadas serán propias de Colombia (o de lugares cercanos), pues la plataforma tiene como principal objetivo operar en Colombia.

```

CONSTRAINT RangoGeográficoColombia CHECK (latitud BETWEEN -4.2 AND 13.5 AND longitud BETWEEN -79.0 AND -66.0)

```

Figura 4.2.5. Script de ejemplo 5

- Los nombres y apellidos pertenecientes a los usuarios son validados mediante la restricción LIKE '%[^0-9]{}'. Esta restricción obliga a las cadenas de texto ingresadas como nombres y apellidos a que no puedan tener números en su composición. Como podrá notar, la restricción es bastante flexible ya que admite cadenas como '._&/'; sin embargo, por simplicidad del script se prefirió mantener una expresión regular simple que restringiera el dominio de manera funcional y a su vez no involucre más aspectos complejos de los necesarios que pueden ser desarrollados en futuras iteraciones.

```

CONSTRAINT NombresVálidos CHECK (TRIM(nombres) <> '' AND nombres NOT LIKE '%[^0-9]{}'),
CONSTRAINT ApellidosVálidos CHECK (TRIM(apellidos) <> '' AND apellidos NOT LIKE '%[^0-9]{}'),

```

Figura 4.2.6. Script de ejemplo 6

- Las tablas que tienen atributos heredados de una tabla padre tienen definida su llave primaria a la vez que como llave foránea de la tabla padre. En otras palabras, las tablas que heredan atributos tienen como PRIMARY KEY una FOREIGN KEY que referencia a su tabla padre.

```
CONSTRAINT PKPerfiles PRIMARY KEY (idPerfil),  
CONSTRAINT FKPerfilesUsuario FOREIGN KEY (idPerfil) REFERENCES Usuarios(idUsuario)
```

Figura 4.2.7. Script de ejemplo 7

- Para validar que una cadena ingresada no esté compuesta de solo espacios se añadió la restricción TRIM(cadena) <> '' a todas aquellas columnas que almacenan texto en la base de datos. Esto permite evitar casos de ingresos erróneos obvios de cadenas de texto

```
CONSTRAINT NombreLocalizaciónVálida CHECK (TRIM(nombre) <> '')
```

Figura 4.2.8. Script de ejemplo 8

- Las fechas ingresadas que deben ser pasadas se verifican con la condición fecha <= GETDATE() que utiliza una función propia del motor SQL Server.

```
CONSTRAINT FechaCreaciónPerfilVálida CHECK (fechaCreación <= GETDATE())
```

Figura 4.2.9. Script de ejemplo 9