Manual Técnico

Versión: 1

Fecha: 22/10/2020

ÍNDICE

1. ANÁLISIS DEL SISTEMA 3
1.1. Objeto 3
1.1.1. Específicos 3
1.2. Introducción 3
2. Requerimientos técnicos 3
2.1. Requerimientos mínimos de Hardware 3
2.2. Requerimientos mínimos de Software 3
3. Herramientas utilizadas para el desarrollo 3
3.1. C# 3
3.2. ASP.NET Core 4
3.3. Entity Framework Core 4
3.4. MySQL 4
3.5. Angular 4
3.6. TypeScript 4
3.7. Apache Cordova 4
3.8. Git 5
4. Descripción de Arquitectura de software 5
4.1. Dominio 5
4.2. Aplicación 5
4.3. Infraestructura 5
4.4. Interfaz 5
5. Configuraciones del sistema 5
5.1. Conexión a base de datos 5
5.2. Configuración de Identity Server 6
5.3. Ejecución 8
6. Diccionario de datos 8

1. ANÁLISIS DEL SISTEMA

1.1. Objeto

Brindar información para personal técnico que quiera conocer sobre las configuraciones y organización del sistema.

1.1.1. Específicos

- Dar una idea general de la organización del proyecto para que pueda ser entendido por personal técnico que necesite realizar modificaciones, etc.
- Describir las configuraciones más importantes que se necesitan para poder ejecutar el proyecto en un entorno local.
- Describir las herramientas tecnológicas utilizadas en el desarrollo del proyecto.
- Describir la estructura de los datos por medio de un diccionario de datos.

1.2. Introducción

Este manual sirve como una guía para todo tipo de personas de nivel técnico que necesiten conocer sobre la organización del proyecto, así como de las tecnologías utilizadas, esto permitirá que sean capaces de entender a un nivel más técnico al sistema y sus configuraciones e incluso que puedan ejecutar el proyecto de forma local. Asimismo, se realiza una descripción detallada de las tablas de la base de datos, para mejorar la comprensión de la estructura de los datos.

2. Requerimientos técnicos

2.1. Requerimientos mínimos de Hardware

Servidor	4 Cores, 7GB de RAM, 10GB de disco duro.
Computadora de desarrollo	4 Cores, 8GB de RAM, 32GB de disco duro.

2.2. Requerimientos mínimos de Software

Sistema operativo	Windows 10

3. Herramientas utilizadas para el desarrollo

3.1. C#

C# es un lenguaje de programación moderno orientado a objetos creado por Microsoft para desarrollar en .NET. Actualmente se encuentra en la versión 8 y entre sus características se encuentra su sencillez, modernidad, seguridad, extensibilidad, etc. Su sintaxis es muy similar a la de Java y al igual que Java compila su código a un lenguaje intermedio para ejecutarlo sobre un runtime.

3.2. ASP.NET Core

ASP.NET Core es un *framework* gratuito y de código abierto para crear aplicaciones web que se ejecutan en .NET Core. Es una mejora de ASP.NET el cual se ejecutaba sobre .NET Framework y por lo tanto sólo se podían crear aplicaciones web que se ejecutan en servidores Windows. Contiene diferentes componentes que se adaptan al tipo de aplicación que se quiere desarrollar

3.3. Entity Framework Core

Entity Framework Core (EF Core) es un ORM de código abierto creado por Microsoft para la plataforma .NET Core. EF Core soporta muchos gestores de bases de datos como MySQL, SQL Server, Oracle, PostgreSQL, etc. a través de *Database Providers* que son implementaciones que permiten interactuar con una base de datos a través de EF Core. EF Core permite a los desarrolladores .NET utilizar objetos de .NET para interactuar con una base de datos SQL

3.4. MySQL

MySQL es una base de datos relacional de código abierto actualmente propiedad de Oracle. MySQL es muy utilizado en aplicaciones web en conjunto con PHP, Apache, y Linux. MySQL utiliza la arquitectura cliente servidor y puede ejecutarse en diferentes sistemas operativos como Linux y Windows. En comparación con otras bases de datos como Oracle, MySQL es fácil de usar y mucho más amigable hacia las personas sin experiencia en bases de datos, lo cual ha contribuido a su popularidad

3.5. Angular

Angular es un *framework* de desarrollo que permite la creación de SPAs complejas y eficientes utilizando HTML y TypeScript. Su objetivo es facilitar el desarrollo de aplicaciones complejas utilizando tecnologías modernas y buenas prácticas de desarrollo que permiten mejorar proceso de creación de SPAs. Angular facilita el desarrollo con MVC e incluye otras funcionalidades como creación de módulos, inyección de dependencias, encapsulamiento de HTML en componentes, entre otros

3.6. TypeScript

TypeScript es un lenguaje pensado para mejorar JavaScript. Permite a los desarrolladores utilizar características no disponibles en JavaScript como el uso de tipos de datos, enums, interfaces, programación genérica, etc. El código TypeScript se compila a JavaScript lo cual permite crear aplicaciones en TypeScript que se ejecuten en los navegadores web, Node.js o cualquier otro motor de ejecución de JavaScript. El principal objetivo de TypeScript es ayudar a detectar errores lo más pronto posible usando tipos de datos y de esta forma que el desarrollo con JavaScript sea más eficiente. Debido al uso de tipos de datos, los IDEs de programación son capaces de ofrecer de mejora manera ayuda a los programadores

3.7. Apache Cordova

Apache Cordova es un framework de desarrollo de aplicaciones móviles multiplataforma que permite la utilización de tecnologías web como HTML, CSS y JavaScript para el desarrollo. Esto permite que las aplicaciones puedan ser portadas fácilmente a los sistemas operativos iOS o Android proveyendo una interfaz consistente en ambas plataformas.

3.8. Git

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

4. Descripción de Arquitectura de software

Para la organización del código del sistema se utilizó la arquitectura Clean Arquitecture implementándola con un *template* para .NET Core que se llama Clean Arquitecture y que se puede encontrar en el siguiente enlace:

https://github.com/jasontaylordev/CleanArchitecture

La arquitectura Clean Arquitecture, permite organizar el código de la siguiente forma:

4.1. Dominio

Este contiene todas las clases que representan entidades, enums, interfaces, excepciones, tipos de datos y toda la lógica del dominio. Representado por el proyecto ubicado en src/Domain.

4.2. Aplicación

Esta parte contiene la lógica de la aplicación. Depende del dominio, pero no tiene dependencias en otros proyectos. Aquí se definen interfaces que son implementadas por otras partes, comúnmente infraestructura. Se representa por el proyecto en serc/Application

4.3. Infraestructura

En esta parte es donde se implementan las interfaces definidas en aplicación para acceder a recursos externos como archivos, servicios web, base de datos, etc. Todo lo relacionado a la infraestructura se encuentra en src/Infrastructure.

4.4. Interfaz

Contiene la parte gráfica del proyecto y hace uso de aplicación e infraestructura. Esta parte se representa con el proyecto src/WebUI que además de contener los servicios web también contiene la aplicación Angular.

Además de Clean Arquitecture, también se utiliza el patrón MVC para la aplicación Angular y los servicios web.

5. Configuraciones del sistema

5.1. Conexión a base de datos

La cadena de conexión a base de datos se encuentra en el archivo src/WebUI/aappsettings.json

```
"ConnectionStrings": {
  "DefaultConnection": "server=192.168.10.10;database=rentasgt;user=rentasgt;password=rentasgt"
},
```

En el archivo src/Infrastructure/DependencyInjection.cs se encuentra la configuración de Entity Framework para que utiliza la cadena de conexión a la base de datos:

```
services.AddDbContext<ApplicationDbContext>(options =>
                    options.UseMySql(
                        configuration.GetConnectionString("DefaultConnection"),
                        b => b.MigrationsAssembly(typeof(ApplicationDbContext).Assembly.FullName)));
```

Las configuraciones individuales de las entidades para Entity Framework se encuentran en la carpeta src/Infrastructure/Persistence/Configurations. Existe una clase por cada entidad, por ejemplo, la siguiente es parte de la configuración para la entidad Product.

```
public class ProductConfiguration : IEntityTypeConfiguration<Product>
    {
        public void Configure(EntityTypeBuilder<Product> builder)
        {
            builder.Property(p => p.Id).ValueGeneratedOnAdd();

            builder.Property(p => p.Status)
                .IsRequired();

            builder.Property(p => p.Name)
                .IsRequired()
                .HasMaxLength(Product.MAX_NAME_LENGTH);
            builder.HasIndex(p => p.Name)
                .IsUnique(false);

            builder.Property(p => p.OtherNames)
                .HasMaxLength(Product.MAX_OTHERNAMES_LENGTH)
                .IsRequired();
            builder.HasIndex(p => p.OtherNames)
                .IsUnique(false);
```

Antes de ejecutar el proyecto, se debe crear la base de datos junto con un usuario para conectarse, además de algunas funciones. Para esto, ejecutar el script SQL que se ubica en src/Infrastructure/Persistence/DB_USERCREATION.sql. Las tablas se crearan automáticamente cuando se ejecute el proyecto.

5.2. Configuración de Identity Server

En el archivo src/WebUI/appsettings.Production.json se encuentra la configuración del certificado SSL para Identity Server:

```
"IdentityServer": {
    "Key": {
      "Type": "Store",
      "StoreName": "My",
      "StoreLocation": "CurrentUser",
      "Name": "CN=rentasguatemala.com"
    }
  }
```

En el archivo src/WebUI/appsettings.json se configura el cliente para la aplicación Angular utilizando el perfil IdentityServerSPA:

```
"IdentityServer": {
    "Clients": {
      "rentasgt.WebUI": {
        "Profile": "IdentityServerSPA"
      }
    }
  },
```

El cliente para la aplicación Android se configura en el archivo src/Infrastructure/DependencyInjection.cs

```
configure.Clients.Add(new Client {
    ClientId = "rentasgt.MobileApp",
    RequirePkce = false,
    AllowedGrantTypes = new List<string> { "authorization_code"},
    AllowedScopes = new List<string> { "openid", "profile", "rentasgt.WebUIAPI" },
    RedirectUris = new List<string> { "com.rentasguatemala://oauth_callback", "rentasgt://callback" },
    PostLogoutRedirectUris = new List<string> { "com.rentasguatemala://oauth_callback", "rentasgt://callba
ck" },
    RequireClientSecret = false,
    RequireConsent = false,
    AllowAccessTokensViaBrowser = true,
});
```

En el archivo src/Infrastructure/DependencyInjection.cs también se configuran las políticas de contraseña de Identity Server:

```
services.AddDefaultIdentity<AppUser>(config => {
                config.Password.RequireDigit = false;
                config.Password.RequireLowercase = false;
                config.Password.RequireUppercase = false;
                config.Password.RequireNonAlphanumeric = false;
                config.Password.RequiredLength = 1;
                config.SignIn.RequireConfirmedAccount = false;
                config.SignIn.RequireConfirmedEmail = true;
})
```

5.3. Ejecución

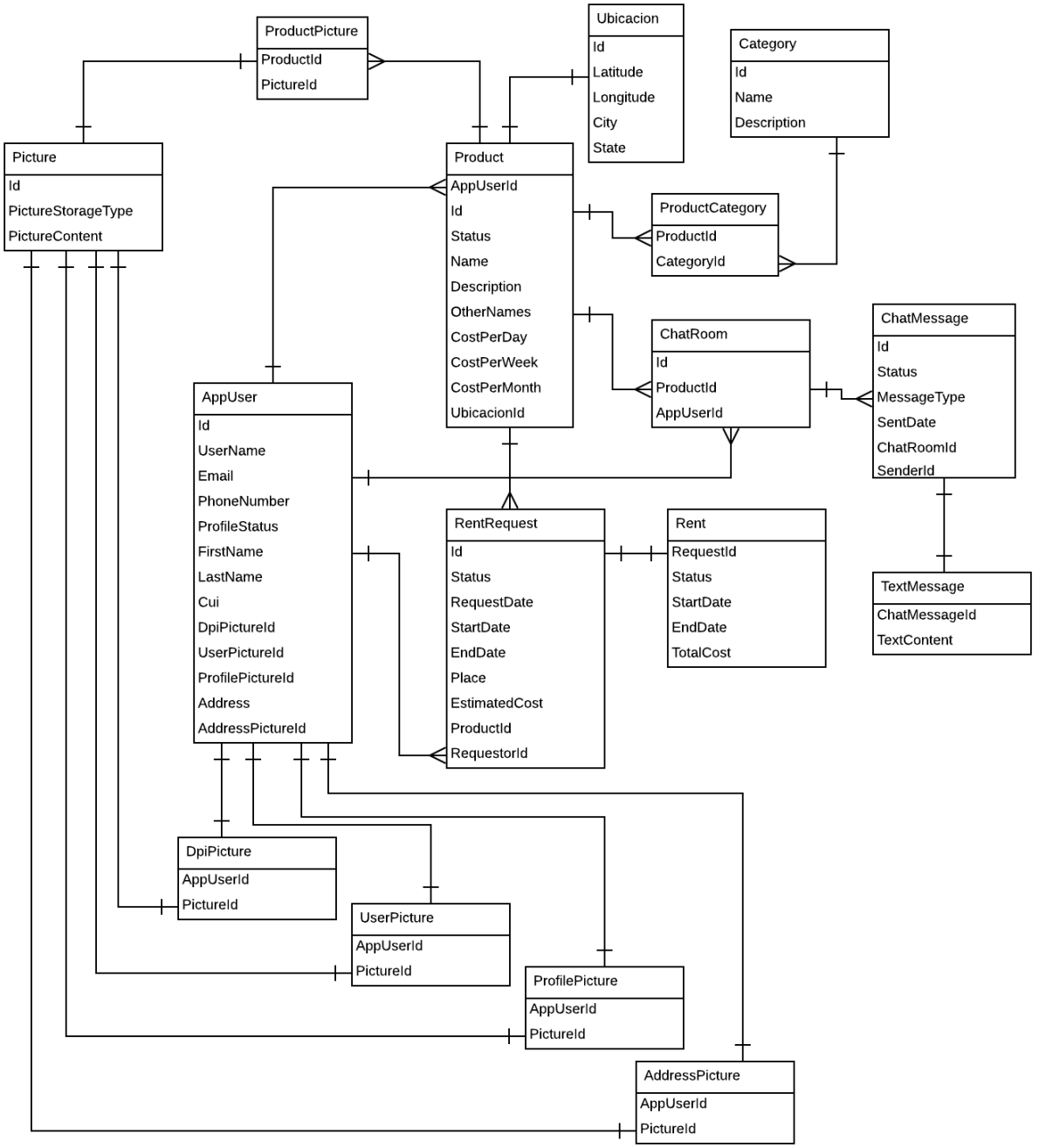
Una vez hechas todas las configuraciones, ya se puede ejecutar el proyecto de forma local con el siguiente comando:

dotnet run --configuration=Debug

Para ver la aplicación, ir a la dirección https://localhost:5001

6. Diccionario de datos

Descripción de las tablas que forman parte del sistema, empezandoe con el diagrama entidad-relación.



Nombre de la tabla: Categories
Descripción: Guarda la información de las categorías disponibles en el sistema.

Campo	Tipo de dato	Tamaño	Descripción
Id	BIGINT	20	Numéro único para identificar al registro.
Name	VARCHAR	128	Nombre de la categoría.
Description	VARCHAR	512	Descripción de la categoría.

CreatedBy	VARCHAR	255	Id del usuario que creó el registro.
Created	DATETIME	6	Fecha y hora de creación del registro.
LastModifiedBy	VARCHAR	255	Id del usuario que modificó el registro por última vez.
LastModified	DATETIME	6	Fecha y hora de la última modificación al registro.

Llave primaria: Id
Relaciones: ProductCategories

Nombre de la tabla: ProductCategories
Descripción: Tabla para representar la relación de muchos a muchos entre artículos y categorías.

Campo	Tipo de dato	Tamaño	Descripción
CategoryId	BIGINT	20	Llave foránea para relacionar con la tabla Categories.
ProductId	BIGINT	20	Llave foránea para relacionar con la tabla Products.

Llave primaria: CategoryId, ProductId
Relaciones: Categories, Products

Nombre de la tabla: Products
Descripción: Almacena la información de los artículos que registran los usuarios.

Campo	Tipo de dato	Tamaño	Descripción
Id	BIGINT	20	Identificador único del registro.
Name	VARCHAR	128	Nombre del artículo
OtherNames	VARCHAR	256	Lista de nombres alternativos del artículo separados por coma.
Description	VARCHAR	512	Texto para describir el artículo.
Status	INT	11	Estado del artículo.
CostPerDay	DECIMAL	65,30	Costo de renta diario del artículo.
CostPerWeek	DECIMAL	65,30	Costo de renta por semana.
CostPerMonth	DECIMAL	65,30	Costo de renta por mes.
Rating	DOUBLE		Calificación del artículo.
Location_Latitude	DOUBLE		Latitud de las coordenadas de ubicación del artículo.
Location_Longitude	DOUBLE		Longitud de las coordenadas de ubicación del artículo.
Location_State	VARCHAR	128	Departamento donde se ubica el artículo.
Location_City	VARCHAR	128	Ciudad donde se ubica el artículo.

Location_StaticMap	LONGTEXT		Ruta de la imagen de la ubicación en el mapa del artículo.
OwnerId	VARCHAR	255	Id del dueño del artículo.
CreatedBy	VARCHAR	64	Id del usuario que creó el registro.
Created	DATETIME	6	Fecha y hora de creación del registro.
LastModifiedBy	VARCHAR	64	Id del usuario que modificó el registro por última vez.
LastModified	DATETIME	6	Fecha y hora de la última modificación al registro.

Llave primaria: Id
Relaciones: AspNetUsers

Nombre de la tabla: AspNetUsers
Descripción: Tabla donde se almacenan los datos de los usuarios del sistema.

Campo	**Tipo de dato**	**Tamaño**	**Descripción**
Id	VARCHAR	255	Identificador único de cada usuario.
UserName	VARCHAR	256	Nombre de usuario.
NormalizedUserName	VARCHAR	256	Nombre de usuario normalizado.
Email	VARCHAR	256	Correo electrónico del usuario.
NormalizedEmail	VARCHAR	256	Correo electrónico normalizado.
EmailConfirmed	TINYINT	1	Indica si el correo electrónico del usuario ya ha sido verificado o no.
ProfileStatus	INT	11	Estado del perfil de usuario.
FirstName	VARCHAR	128	Nombres del usuario.
LastName	VARCHAR	128	Apellidos del usuario.
Cui	CHAR	13	Código Único de Identificación.
ValidatedDpi	TINYINT	1	Almacena si el DPI ya fue validado o no.
Address	VARCHAR	256	Dirección de domicilio del usuario.
ValidatedAddress	TINYINT	1	Indica si la dirección ya fue verificada o no.
Reputation	DOUBLE		Calificación del usuario.
PhoneNumber	VARCHAR	16	Número de teléfono del usuario.

PhoneNumberConfirmed	TINYINT	1	Indica si el número de teléfono ya ha sido verificado o no.
PasswordHash	LONGTEXT		Hash de la contraseña del usuario.
TwoFactorEnabled	TINYINT	1	Indica si el usuario tiene habilitada la autenticación por dos factores.
LockoutEnabled	TINYINT	1	Indica si la cuenta ha sido bloqueada.
LockoutEnd	DATETIME	6	Fecha y hora de cuándo se puede volver a utilizar la cuenta después de haber sido bloqueada.
AccessFailedCount	INT	11	Contador de la cantidad de veces que ha intentado acceder sin éxito.
CreatedBy	VARCHAR	64	Id del usuario que creó el registro.
Created	DATETIME	6	Fecha y hora de creación del registro.
LastModifiedBy	VARCHAR	64	Id del usuario que modificó el registro por última vez.
LastModified	DATETIME	6	Fecha y hora de la última modificación al registro.

Llave primaria: Id
Relaciones:

Nombre de la tabla: RentRequests
Descripción: Datos de las solicitudes de renta de artículos realizadas por los usuarios.

Campo	**Tipo de dato**	**Tamaño**	**Descripción**
Id	BIGINT	20	Identificador único del registro.
Status	INT	11	Estado de la solicitud.
RequestDate	DATETIME	6	Fecha y hora en que se realizó la solicitud.
StartDate	DATETIME	6	Fecha en la que el solicitante quiere iniciar la renta.
EndDate	DATETIME	6	Fecha en la que el solicitante quiere terminar la renta.
ProductId	BIGINT	20	Llave foránea del producto que el solicitante quiere rentar.
RequestorId	VARCHAR	255	Id del usuario que creó la solicitud.
EstimatedCost	DECIMAL	65,30	Costo estimado de la renta.

Place	VARCHAR	128	Lugar de reunión sugerido por el solicitante.
CreatedBy	VARCHAR	64	Id del usuario que creó el registro.
Created	DATETIME	6	Fecha y hora de creación del registro.
LastModifiedBy	VARCHAR	64	Id del usuario que modificó el registro por última vez.
LastModified	DATETIME	6	Fecha y hora de la última modificación al registro.

Llave primaria:
Relaciones:

Nombre de la tabla: RequestEvents
Descripción: Tabla para almacenar información de los eventos relacionados con una solicitud de renta.

Campo	Tipo de dato	Tamaño	Descripción
Id	BIGINT	20	Identificador único del evento.
EventType	INT	11	Tipo de evento.
EventDate	DATETIME	6	Fecha y hora del evento.
Message	VARCHAR	128	Un mensaje o descripción del evento.
RentRequestId	BIGINT	20	Llave foránea de la solicitud a la que pertenece el evento.

Llave primaria: Id
Relaciones: RentRequests

Nombre de la tabla: Rents
Descripción: Almacena información de las rentas que se realizan en el sistema.

Campo	Tipo de dato	Tamaño	Descripción
RequestId	BIGINT	20	Id de la renta y llave foránea de la solicitud que creó la renta.
Status	INT	11	Estado de la renta.
StartDate	DATETIME	6	Fecha y hora en que inició la renta.
EndDate	DATETIME	6	Fecha y hora en que terminó la renta.
TotalCost	DECIMAL	65,30	Costo de la renta.
CreatedBy	VARCHAR	64	Id del usuario que creó el registro.
Created	DATETIME	6	Fecha y hora de creación del registro.

LastModifiedBy	VARCHAR	64	Id del usuario que modificó el registro por última vez.
LastModified	DATETIME	6	Fecha y hora de la última modificación al registro.

Llave primaria: RequestId
Relaciones:

Nombre de la tabla: ChatRooms
Descripción: Tabla que guarda la información de las salas de chat. Es una tabla que sirve para relacionar los mensajes que se envían los usuarios con respecto a un artículo.

Campo	Tipo de dato	Tamaño	Descripción
Id	BIGINT	20	Identificador único del registro.
ProductId	BIGINT	20	Llave foránea del artículo.
UserId	VARCHAR	255	Llave foránea del usuario.
LastMessageId	BIGINT	20	Llave foránea del último mensaje que se envío en la sala.

Llave primaria: Id
Relaciones: AspNetUsers, Products, ChatMessages

Nombre de la tabla: ChatMessages
Descripción: En esta tabla se almacenan los mensajes que se envían los usuarios.

Campo	Tipo de dato	Tamaño	Descripción
Id	BIGINT	20	Identificador único del mensaje.
MessageType	INT	11	Tipo de mensaje.
SentDate	DATETIME	6	Fecha y hora en que se envió el mensaje.
Content	VARCHAR	512	Contenido del mensaje.
RoomId	BIGINT	20	Llave foránea de la sala de chat a la que pertenece el mensaje.
SenderId	VARCHAR	255	Llave foránea del usuario que envió el mensaje.

Llave primaria: Id
Relaciones: ChatRooms, AspNetUsers

Nombre de la tabla: Conflicts
Descripción: Tabla que almacena información sobre los conflictos entre usuarios.

Campo	Tipo de dato	Tamaño	Descripción
Id	BIGINT	20	Identificador único del conflicto.
Status	INT	11	Estado del conflicto.
Description	VARCHAR	1024	Descripción del conflicto.
ConflictDate	DATETIME	6	Fecha y hora en que se creó el conflicto.

RentId	BIGINT	20	Llave foránea de la renta en la que se dio el conflicto.
ComplainantId	VARCHAR	255	Llave foránea del usuario que creó el conflicto.
ModeratorId	VARCHAR	255	Llave foránea del usuario encargado de moderar en el conflicto.
CreatedBy	VARCHAR	64	Id del usuario que creó el registro.
Created	DATETIME	6	Fecha y hora de creación del registro.
LastModifiedBy	VARCHAR	64	Id del usuario que modificó el registro por última vez.
LastModified	DATETIME	6	Fecha y hora de la última modificación al registro.

Llave primaria: Id
Relaciones: Rents, AspNetUsers

Nombre de la tabla: ConflictRecords
Descripción: Tabla donde se guardan las anotaciones que hace un moderador sobre un conflicto.

Campo	Tipo de dato	Tamaño	Descripción
Id	BIGINT	20	Identificador único del registro.
ConflictId	BIGINT	20	Llave foránea del conflicto al que pertenece la anotación.
Description	VARCHAR	1024	Descripción.
RecordDate	DATETIME	6	Fecha y hora de creación de la anotación.

Llave primaria: Id
Relaciones: Conflicts

Nombre de la tabla: Pictures
Descripción: Tabla para almacenar imágenes.

Campo	Tipo de dato	Tamaño	Descripción
Id	BIGINT	20	Identificador único de la imagen.
StorageType	INT	11	Tipo de almacenamiento de la imagen (Base64 o URL).

PictureContent	LONGTEXT		Contenido de la imagen que puede ser una URL o cadena Base64.

Llave primaria: Id
Relaciones:

Nombre de la tabla: ProductPictures
Descripción: Tabla para relacionar productos con sus imágenes.

Campo	Tipo de dato	Tamaño	Descripción
ProductId	BIGINT	20	Llave foránea del producto al que pertenece la imagen.
PictureId	BIGINT	20	Llave foránea de la imagen.

Llave primaria: UserId, PictureId
Relaciones: Products, Pictures

Nombre de la tabla: AddressPictures
Descripción: Tabla para relacionar a un usuario con una imagen para validar la dirección.

Campo	Tipo de dato	Tamaño	Descripción
UserId	VARCHAR	255	Llave foránea del usuario al que pertenece la imagen.
PictureId	BIGINT	20	Llave foránea de la imagen.

Llave primaria: UserId
Relaciones: AspNetUsers, Pictures

Nombre de la tabla: DpiPictures
Descripción: Tabla para relacionar a un usuario con la imagen de su DPI.

Campo	Tipo de dato	Tamaño	Descripción
UserId	VARCHAR	255	Llave foránea del usuario al que pertenece la imagen.
PictureId	BIGINT	20	Llave foránea de la imagen.

Llave primaria: UserId
Relaciones: AspNetUsers, Pictures

Nombre de la tabla: UserPictures
Descripción: Tabla para relacionar a un usuario con su foto.

Campo	Tipo de dato	Tamaño	Descripción
UserId	VARCHAR	255	Llave foránea del usuario al que pertenece la imagen.
PictureId	BIGINT	20	Llave foránea de la imagen.

Llave primaria: UserId
Relaciones: AspNetUsers, Pictures

Nombre de la tabla: AddressPictures
Descripción: Tabla para relacionar a un usuario con una imagen para su perfil.

Campo	Tipo de dato	Tamaño	Descripción
UserId	VARCHAR	255	Llave foránea del usuario al que pertenece la imagen.
PictureId	BIGINT	20	Llave foránea de la imagen.

Llave primaria: UserId
Relaciones: AspNetUsers, Pictures

Nombre de la tabla: RatingToProducts
Descripción: Tabla donde se almacenan las calificaciones que le han dado los usuarios a un artículo y a su dueño.

Campo	Tipo de dato	Tamaño	Descripción
Id	BIGINT	20	Identificador único del registro.
Status	INT	11	Estado de la calificación.
RatingDate	DATETIME	6	Fecha y hora en que se dio la calificación.
FromUserId	VARCHAR	255	Llave foránea del usuario que da la calificación.
ProductId	BIGINT	20	Llave foránea del producto al que recibe la calificación.
ProductRatingValue	INT	11	Calificación al artículo.
OwnerRatingValue	INT	11	Calificación al dueño del artículo.
Comment	VARCHAR	1024	Comentario al artículo.

Llave primaria: Id
Relaciones: Products, AspNetUsers

Nombre de la tabla: RatingToUsers
Descripción: Tabla donde se almacenan las calificaciones recibidas por los usuarios que rentan.

Campo	Tipo de dato	Tamaño	Descripción
Id	BIGINT	20	Identificador único del registro.
Status	INT	11	Estado de la calificación.
RatingValue	INT	11	Calificación al usuario.
Comment	VARCHAR	1024	Comentario al usuario.
RatingDate	DATETIME	6	Fecha y hora en que se dio la calificación.
FromUserId	VARCHAR	255	Llave foránea del usuario que da la calificación.
ToUserId	VARCHAR	255	Llave foránea del usuario que recibe la calificación.

Llave primaria: Id
Relaciones: AspNetUsers

Nombre de la tabla: AspNetRoles
Descripción: Roles de usuario.

Campo	Tipo de dato	Tamaño	Descripción
Id	VARCHAR	255	Identificador único del rol.
Name	VARCHAR	256	Nombre del rol.
NormalizedName	VARCHAR	256	Nombre normalizado del rol.

Llave primaria: Id
Relaciones:

Nombre de la tabla: AspNetUserRoles
Descripción: Tabla para asignar roles a usuarios.

Campo	Tipo de dato	Tamaño	Descripción
UserId	VARCHAR	255	Llave foránea del usuario.
RoleId	VARCHAR	255	Llave foránea del rol.

Llave primaria: UserId, RoleId
Relaciones: AspNetRoles, AspNetUsers