Hosting

The database for this project is hosted on Azure at: https://redriver.database.windows.net (https://redriver.database.windows.net)

Choice of Database and Table Creation

At present the RedRiver server uses a Microsoft SQL relational database for all data storage. This has the advantage of playing nicely together with ASP.NET and is the default choice in this context. The Identity framework, together with the Entity framework used in ASP.NET creates its own set of tables in a database, which stem from the ApplicationUser class. ApplicationUser inherits from IdentityUser and so has more fields than shown here.

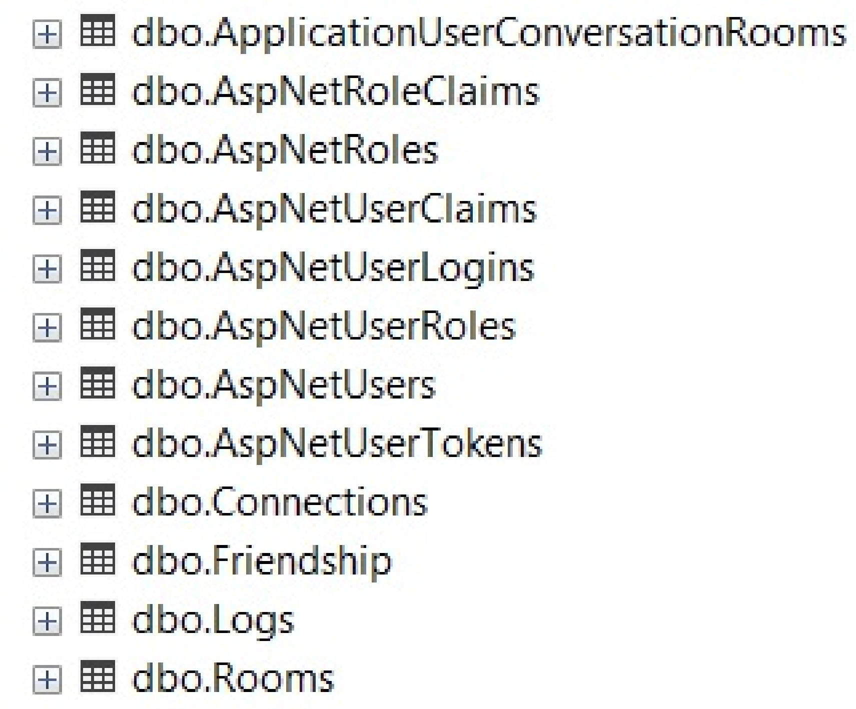
```
public class ApplicationUser : IdentityUser
    {
        public ApplicationUser()
        {
            this.Friendships = new HashSet<Friendship>();
        }
        public string SocialSecurity { get; set; }
        public string StreetAddress { get; set; }
        public string City { get; set; }
        public string Country { get; set; }
        public string Postcode { get; set; }
        public string FirstName { get; set; }
        public string Surname { get; set; }
        public string RelativeUsername { get; set; }
        public string TelephoneNumber { get; set; }
        public virtual ICollection<Friendship> Friendships { get; set; }
    }
```

ApplicationUser contains an ICollection (implemented as a HashSet) which is designed to represents Friendships. The Friendship class is as follows:

```
 public class Friendship
    {
        public int FriendshipId { get; set; }
        public string FriendUsername { get; set; }
        public string FriendId { get; set; }
        [ForeignKey("ApplicationUserId")]
        public string ApplicationUserId { get; set; }
        public DateTime Time { get; set; } = DateTime.Now;
    }
```

Generated Tables

Entity Framework automatically creates the following tables upon system start if they do not already exist.



dbo.ApplicationUserConversationRooms

dbo.AspNetRoleClaims

dbo.AspNetRoles

dbo.AspNetUserClaims

dbo.AspNetUserLogins

dbo.AspNetUserRoles

dbo.AspNetUsers

dbo.AspNetUserTokens

dbo.Connections

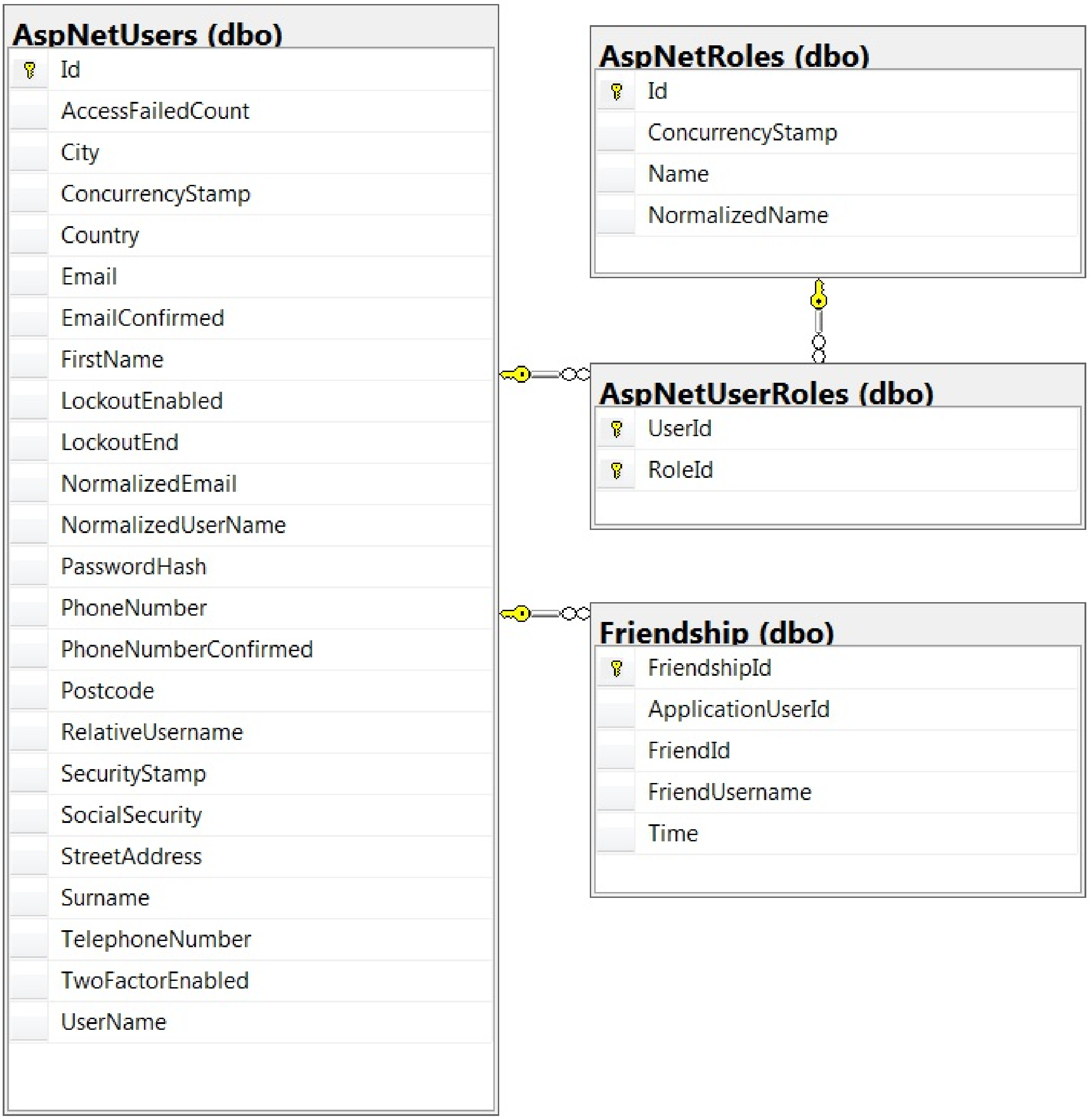
dbo.Friendship

dbo.Logs

dbo.Rooms

Below is a closer look at the AspNetUsers table, which reflects the fields implemented in the ApplicationUser class. Not all tables relating to the user table are shown for the sake of clarity. Users roles for **admin** and **superuser** are available, and the diagram shows the join table between

NetUsers and NetRoles.



AspNetUsers (dbo)
- Id
- AccessFailedCount
- City
- ConcurrencyStamp
- Country
- Email
- EmailConfirmed
- FirstName
- LockoutEnabled
- LockoutEnd
- NormalizedEmail
- NormalizedUserName
- PasswordHash
- PhoneNumber
- PhoneNumberConfirmed
- Postcode
- RelativeUsername
- SecurityStamp
- SocialSecurity
- StreetAddress
- Surname
- TelephoneNumber
- TwoFactorEnabled
- UserName

AspNetRoles (dbo)
- Id
- ConcurrencyStamp
- Name
- NormalizedName

AspNetUserRoles (dbo)
- UserId
- RoleId

Friendship (dbo)
- FriendshipId
- ApplicationUserId
- FriendId
- FriendUsername
- Time

Improvements

This set of classes does not represent the best choice for a relational database since it leads to data duplication; a "friend" in this context is actually another user . FriendUserName is then information which is also stored elsewhere and could be determined by a lookup. However, it is convenient at present. A better choice would be an ICollection Friends field in ApplicationUser. This was difficult to implement and the set of classes shown above were chosen instead. **Going forward, this should be corrected before the system is set into a production environment.**

Similarly, when two users become friends two entries are made in the Friendships table i.e. a friendship is currently one way only. There is possibly room for improvement in this system since friendships in practice are always mutual.

Security

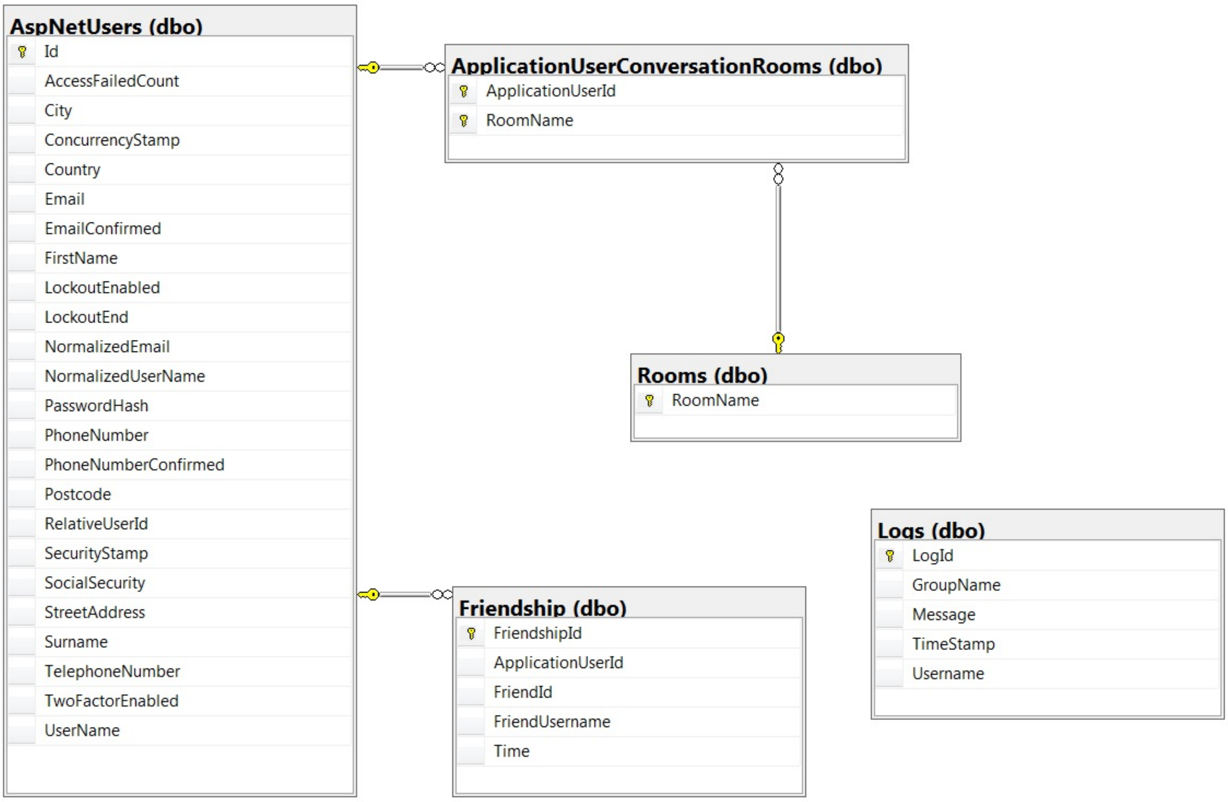
The chosen database is secure in the sense that Azure provides the option of encrypting all information in the database. While this may seem desirable, it may not provide full security since the encryption methods are not known and the ability for Microsoft to withstand legal challenges from government agencies is unclear.

ToDo

Further database tables will be required in accordance with the functional requirements. It is possible that friendship requests will have to be stored, necesitating a table. A table for logging will also be required.

Update 20180502

Tables have been added to the database to cope with group chat and logs. A group chat is referred to as a "room" within the database.



AspNetUsers (dbo)
Id
AccessFailedCount
City
ConcurrencyStamp
Country
Email
EmailConfirmed
FirstName
LockoutEnabled
LockoutEnd
NormalizedEmail
NormalizedUserName
PasswordHash
PhoneNumber
PhoneNumberConfirmed
Postcode
RelativeUserId
SecurityStamp
SocialSecurity
StreetAddress
Surname
TelephoneNumber
TwoFactorEnabled
UserName
ApplicationUserConversationRooms (dbo)
ApplicationUserId
RoomName
Rooms (dbo)
RoomName
Logs (dbo)
LogId
GroupName
Message
TimeStamp
Username
Friendship (dbo)
FriendshipId
ApplicationUserId
FriendId
FriendUsername
Time