This page contains documentation concerning technical aspects of the RedRiver server.

The RedRiver server is primarily designed as an api server, and so the client and server communicate through API calls. However, the messaging system uses SignalR.

Frameworks and Architecture

The base server implemention uses ASP.NET Core 2. This has the advantage of being compatible with the majority of operating systems, in contrast to earlier ASP.NET variants.

An MVC architecture is inherent to the ASP.NET framework and this has been followed in order to allow for separation of concerns. However, since we have an external client which is not served from ASP.NET, the V (View) in MVC has no relevance in this particular context.

Technology

- HTTP requests and responses - ASP.NET HTTP (https://docs.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.http?view=aspnetcore-2.0)
- Websockets - SignalR (https://www.asp.net/signalr)
- Video call and streaming - WebRTC (https://webrtc.org/start/)
- Authentication/Authorization - JWT (https://jwt.io/introduction/)

Hosting

At present the server is hosted as an Azure webapp. (https://serverredriver.azurewebsites.net/api/routetest/tryunprotectedroute)
App generation and configuration can be configured through the Azure Portal (https://portal.azure.com), although the particular logins necessary for this project are not published here.

Deployment

All changes to the master branch of the repository are automatically pushed to the deployment server on Azure.

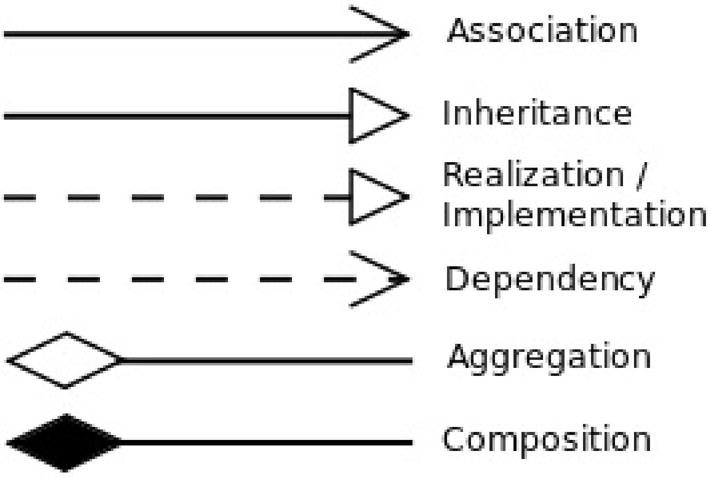
Installation

Installation instructions are found in the repo's README.md. (https://github.com/jimmybengtsson/grupp03-redriver/blob/master/server/README.md)

Architecture

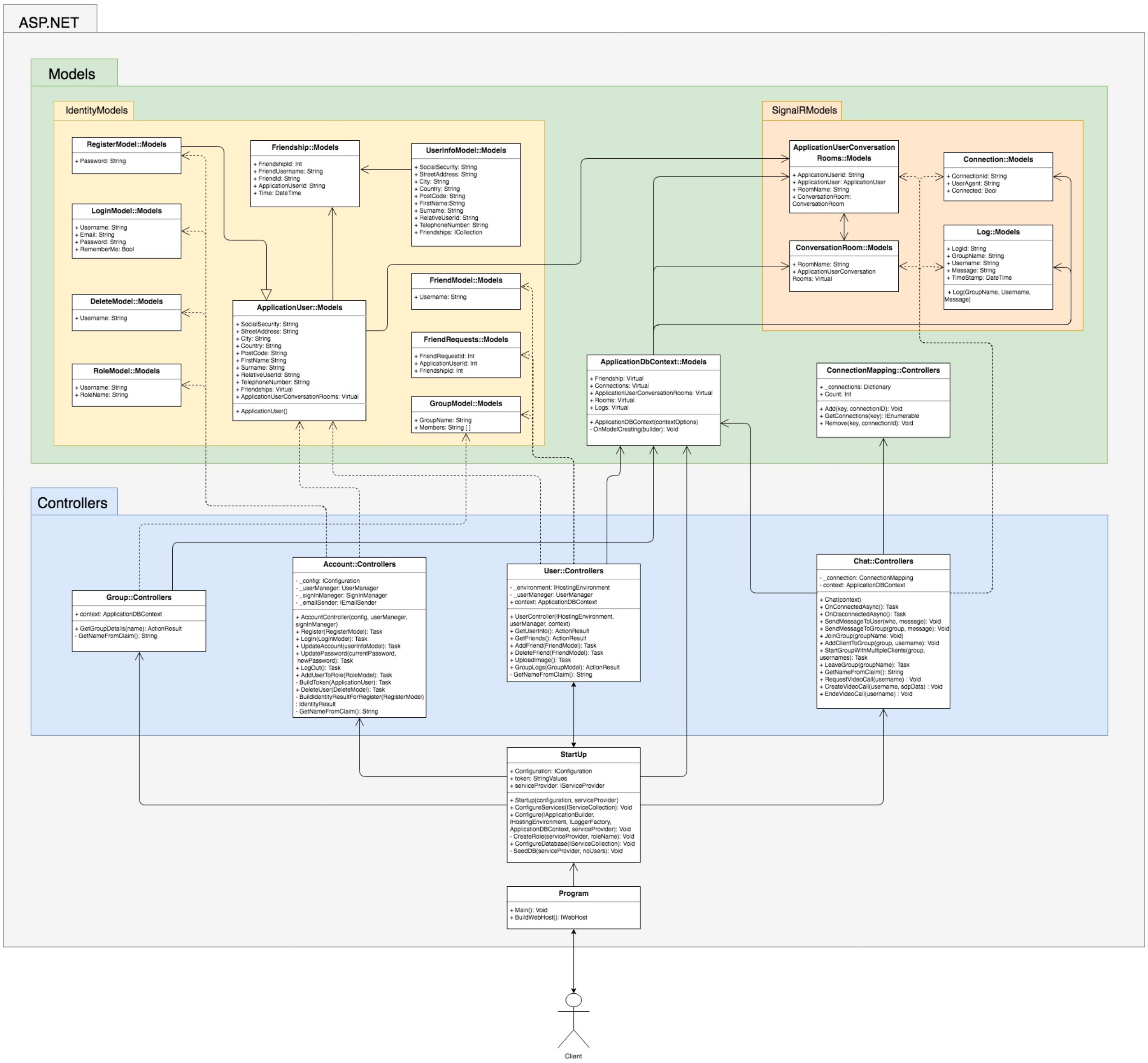
To show the structure of the application/system we use a UML class diagram. A class diagram is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, operations (or methods), and the relationships among objects.

The arrows represents the relationship between classes:



For more information about the notation used click here: Wikipedia (https://en.wikipedia.org/wiki/Class_diagram)

Full size (https://github.com/jimmybengtsson/grupp03-redriver/blob/master/documentation/diagrams/png/Server-Architecture-v4.png)


ASP.NET
Models
IdentityModels
RegisterModel::Models
+ Password: String
LoginModel::Models
+ Username: String
+ Email: String
+ Password: String
+ RememberMe: Bool
DeleteModel::Models
+ Username: String
RoleModel::Models
+ Username: String
+ RoleName: String
Friendship::Models
+ FriendshipId: Int
+ FriendUsername: String
+ FriendId: String
+ ApplicationUserId: String
+ Time: DateTime
UserInfoModel::Models
+ SocialSecurity: String
+ StreetAddress: String
+ City: String
+ Country: String
+ PostCode: String
+ FirstName:String
+ Surname: String
+ RelativeUserId: String
+ TelephoneNumber: String
+ Friendships: ICollection
ApplicationUser::Models
+ SocialSecurity: String
+ StreetAddress: String
+ City: String
+ Country: String
+ PostCode: String
+ FirstName:String
+ Surname: String
+ RelativeUserId: String
+ TelephoneNumber: String
+ Friendships: Virtual
+ ApplicationUserConversationRooms: Virtual
+ ApplicationUser()
FriendModel::Models
+ Username: String
FriendRequests::Models
+ FriendRequestId: Int
+ ApplicationUserId: Int
+ FriendshipId: Int
GroupModel::Models
+ GroupName: String
+ Members: String [ ]
SignalRModels
ApplicationUserConversation Rooms::Models
+ ApplicationUserId: String
+ ApplicationUser: ApplicationUser
+ RoomName: String
+ ConversationRoom: ConversationRoom
ConversationRoom::Models
+ RoomName: String
+ ApplicationUserConversation Rooms: Virtual
Connection::Models
+ ConnectionId: String
+ UserAgent: String
+ Connected: Bool
Log::Models
+ LogId: String
+ GroupName: String
+ Username: String
+ Message: String
+ TimeStamp: DateTime
+ Log(GroupName, Username, Message)
ApplicationDbContext::Models
+ Friendship: Virtual
+ Connections: Virtual
+ ApplicationUserConversationRooms: Virtual
+ Rooms: Virtual
+ Logs: Virtual
+ ApplicationDBContext(contextOptions)
- OnModelCreating(builder): Void
ConnectionMapping::Controllers
+ _connections: Dictionary
+ Count: Int
+ Add(key, connectionID): Void
+ GetConnections(key): IEnumerable
+ Remove(key, connectionId): Void
Controllers
Group::Controllers
+ context: ApplicationDBContext
+ GetGroupDetails(name): ActionResult
- GetNameFromClaim(): String
Account::Controllers
- _config: IConfiguration
- _userManeger: UserManager
- _signInManeger: SignInManager
- _emailSender: IEmailSender
+ AccountController(config, userManeger, signInManeger)
+ Register(RegisterModel): Task
+ Login(LoginModel): Task
+ UpdateAccount(userInfoModel): Task
+ UpdatePassword(currentPassword, newPassword): Task
+ LogOut(): Task
+ AddUserToRole(RoleModel): Task
- BuildToken(ApplicationUser): Task
+ DeleteUser(DeleteModel): Task
- BuildIdentityResultForRegister(RegisterModel) : IdentityResult
- GetNameFromClaim(): String
User::Controllers
- _environment: IHostingEnvironment
- _userManeger: UserManager
+ context: ApplicationDBContext
+ UserController(IHostingEnvironment, userManager, context)
+ GetUserInfo(): ActionResult
+ GetFriends(): ActionResult
+ AddFriend(FriendModel): Task
+ DeleteFriend(FriendModel): Task
+ UploadImage(): Task
+ GroupLogs(GroupModel): ActionResult
- GetNameFromClaim(): String
Chat::Controllers
- _connection: ConnectionMapping
- context: ApplicationDBContext
+ Chat(context)
+ OnConnectedAsync(): Task
+ OnDisconnectedAsync(): Task
+ SendMessageToUser(who, message): Void
+ SendMessageToGroup(group, message): Void
+ JoinGroup(groupName: Void)
+ AddClientToGroup(group, username): Void
+ StartGroupWithMultipleClients(group, usernames): Task
+ LeaveGroup(groupName): Task
+ GetNameFromClaim(): String
+ RequestVideoCall(username) : Void
+ CreateVideoCall(username, sdpData) : Void
+ EndeVideoCall(username) : Void
StartUp
+ Configuration: IConfiguration
+ token: StringValues
+ serviceProvider: IServiceProvider
+ Startup(configuration, serviceProvider)
+ ConfigureServices(IServiceCollection): Void
+ Configure(IApplicationBuilder, IHostingEnvironment, ILoggerFactory, ApplicationDBContext, serviceProvider): Void
- CreateRole(serviceProvider, roleName): Void
+ ConfigureDatabase(IServiceCollection): Void
- SeedDB(serviceProvider, noUsers): Void
Program
+ Main(): Void
+ BuildWebHost(): IWebHost
Client