Models:
I models mappen finns alla modeller som används. Det finns rena modeller(Group, Message, IdentityModels) och ViewModels(de som har ViewModel i namnet. De rena modellerna är de enda klasserna som har direkt åtkomst till databasen.

Vissa av modellerna som Message har metoder som returnerar ViewModel som används när man vill skicka meddelanden till View i lämpligt format. På detta vis skärmar vi av modellklasserna från viewdelarna.

I Controller delen finns metoder för varje View del som hanterar kommunikation mellan view<->controller<->modell. Anrop från view sker till controller som i sin tur hämtar relevant data från modellerna och packar om till viewmodels.

Views är delad i olika sektioner: Account, Groups, Home, Manage, Messages och Shared. Account har genererats av Visual Studio först och främst för registrering i vår applikation. Groups innehåller sidor som används som hör till grupper, index och send är de som vi först och främst använder oss av i applikationen. Home innehåller första sidan där man kan se användardetaljer som senaste inloggningen, antal olästa meddelanden ect. Samma princip används för Messages.

Vi har använt oss av template för Identity och modifierat den för att göra restriktioner på alla sidor och den redirectar till inloggning när man vill ha åtkomst till applikationen.

Vi har all åtkomst till databasen via Modellklasserna. Dessa är optimerade i alla fall då vi accessar databasen med eager/lazy/implicit loading. I de flesta falla används eager och implicit loading, speciellt för meddelanden då meddelanden innehåller virtuell attribut som är relaterad till ApplicationUser(To i Message). Ytterligare optimisering hade kunnat göras om vi hade tänkt på det i början och endast inkluderat id i form av sträng istället för ApplikationUser i Message klassen. Då hade man kommit undan med Lazy loading i många fall där vi använder oss av eager/implicit loading.