Diagrammen

## Class diagram

Ik zal het UI-framework flutter gebruiken om de app te ontwikkelen. Ik zou een state-manager genaamd **provider** willen gebruiken om de applicatie een beter structuur te geven en de UI gedeelte van de logica te scheiden. Dat is de reden waarom de class diagram een **ManagerProvider** heeft. Ieder class heeft zijn eigen services, dat is waar de API calls en andere functies uitgevoerd zullen worden. De **ManagerProvider** zal gebruikt worden om veranderingen aan de UI door te geven.

Provider: <https://docs.flutter.dev/data-and-backend/state-mgmt/simple>

## Database diagram en backend

In de database diagram zijn er 3 many-to-many relaties te vinden, **user\_task** zodat meerdere taken aan meerdere gebruikers kunnen worden gekoppeld, **foodtype\_animal** zodat er een overzicht is van welk voer soort voor welk dier geschikt is en **user\_message** zodat er meerdere berichten naar meerdere gebruikers kan worden gestuurd. Ik dacht ook aan een **task\_animal** zodat er overzicht is van taken die aan een bepaald dier uitgevoerd zijn.

Daarnaast zijn er enkele one-to-many-relaties, zoals tussen **task** en **taskType**. Het lijkt me nuttig om taken van elkaar te onderscheiden en te filteren door ze een type te geven. Voorbeelden van **taskTypes** zijn 'cleaning' of 'feeding'. Er is ook een relatie tussen **stock** en **foodType**, zodat duidelijk is welk type voer in elke voorraad zit. Voorbeelden van **foodTypes** zijn 'wheat' en 'hay'. Ten slotte is er een relatie tussen **animal** en **animalType**, om aan te geven tot welke soort een dier behoort, een voorbeeld van een **animalType** is“horse” of “pig”.

Voor de backend zou ik een API willen maken die informatie uit zal wisselen tussen het app en de database. Voor de API zou ik **FastAPI** willen gebruiken, het is een python framework die gemaakt is om API’s te ontwikkelen en waar ik al enige ervaring mee heb.

FastAPI: <https://fastapi.tiangolo.com/>