# Explanation of the Data Access Layer

As database for this project, we are using firestore by firebase. This was chosen because it offers an easy-to-use database solution which is highly integrable with Android. It provides the capability to add “EventListeners” to documents and collections in the database, which will trigger events in the code. This provides us with a sort of observer pattern, where the individual devices subscribed to the database can receive new data, when it is changed in the database.

The Data Access Layer is split into 2 parts:

**fireStoreCommunicator**: handles the lowest level of communication with the database. This includes adding new lists, adding items to list, adding users to database, subscribing to fridges, receiving data, making lists from this data, etc…

**ServiceUpdater:** this serves as the Programming Interface throughout the app. The service also holds a local copy of all fridges the user has subscribed to. All actions done in the activities/fragments goes through the service. For instance, when adding an item to a list in an activity, the service checks if an item with matching name is already on the list. If so, it increments the quantity of that item in the database, instead of adding a duplicate item.

From the service, we send out a broadcast whenever the service has retrieved new data from the database. This way, the activities/fragments can retrieve the newest data from the service, and thus stay in sync with the database.

The reason for choosing the approach where the service holds a local list of the data in the database is because we had experience with this approach from the Assignment 2.

One could argue that it is bad programming practice to have such a huge amount of logic in these 2 classes, and that we instead should have divided the functionality into several lesser classes. Had we had more time, a rework of the architecture would be a topic of high priority.