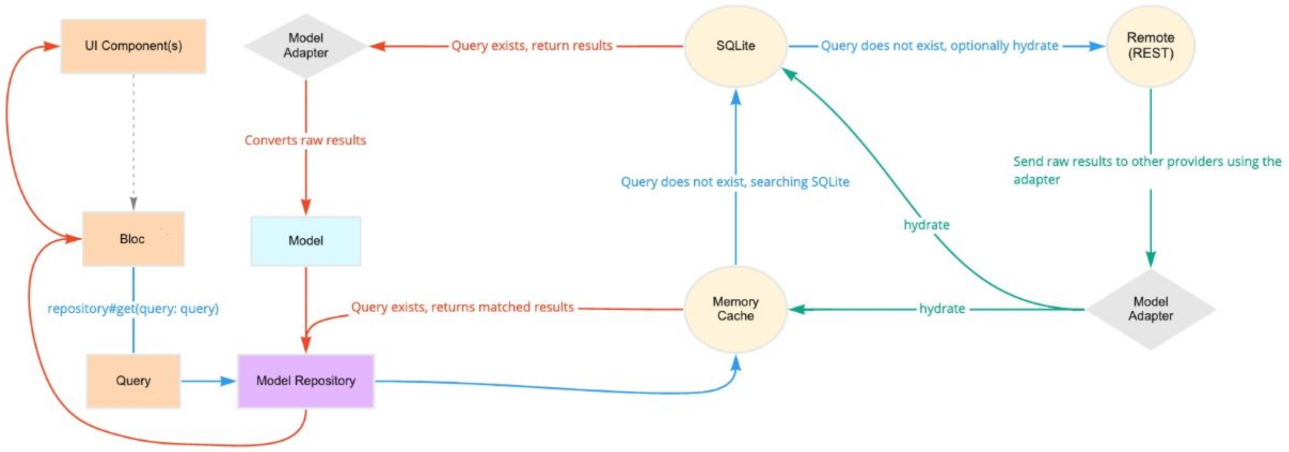
Issue #9

Flutter resources:
https://github.com/tekartik/sqflite/blob/master/sqflite/README.md
https://pub.dev/packages/connectivity

Our solution suggestion

We propose what is called an offline first approach because if Giraf wants to be a widely used app on different app markets, it is expected to be highly functional without internet connection. Offline first also consumes less battery and data - where especially long battery time is important for institutions and families going on trips.


UI Component(s)
Model Adapter
Query exists, return results
SQLite
Query does not exist, optionally hydrate
Remote (REST)
Converts raw results
Query does not exist, searching SQLite
Send raw results to other providers using the adapter
hydrate
Bloc
Model
repository#get(query: query)
Query exists, returns matched results
Memory Cache
hydrate
Model Adapter
Query
Model Repository

Link to bigger picture:
https://cdn.discordapp.com/attachments/674940529124966402/691622267402190858/unknown.png

This model describes how offline first can be done with the bloc pattern.

Whenever the UI needs data it first communicates with the bloc, where the bloc then sends a query to get data. If the data is already in memory we can just get the data, if it is not we try to find it in our local database. If it is not in the local database either we look in the online database where it will be found. When you get data from the online database you also save it in the local database and in memory, such you can have fast retrieval and also data available if you are offline.

The only thing missing the current implementation is for the local database to be implemented with appropriate model adapters. This setup would allow you to read from the local database when offline, however in order to edit, delete or put you would need to store a record of these actions such that you can execute them when online again. To check whether the client is currently in an offline or online state and listen for changes between those states the **connectivity** package can be used. The local database could be implemented either as part of the week planner app or the api client. For the database we propose the **SQFLite** flutter package, since it is the most suggested.

For Giraf it might makes sense to limit the local database to a single user. For the pictograms you would also have to figure out some system where only the most used ones are saved.

Issues / considerations

Before even starting to implement the features on the prioritized list, it is needed to be able to login on an offline device.
One solution could be a setting, where when you are logged in you can allow the current device to be used in offline mode and the user is saved locally, so it is possible to login with no internet connection.

Here is the prioritized list for the offline features

1. Citizen features: Limited to current week.
   1.1 View weekplan.
   1.2 View activity.
   1.3 Mark activity as completed.
   1.4 Timer functionality.
2. Guardian features: Limited to current week.
   2.1 View weekplan.
   2.2 View activity.
   2.3 Cancel activity.
   2.4 Timer functionality.
   2.5 Edit weekplan.
3. Guardian features:
   3.1 Take picture as pictogram.
   3.2 Create/delete weekplans.
   3.3 All functions from point 2 just not limited to the current week.

To store the data (activities, timers) locally on the device the SQLite DB can be used. SQLite is a database that is running on the phone/tablet already. It has a plugin for Flutter (sqflite) that supports both iOS and Android.

Syncing the local database to match the online database:
A consideration could be if some data is more important to sync than other and then make different sync cycles or prioritization, in case the user only has connection for a short period of time. It could also be considered if all data needs to be updated all the time or some data just need daily, weekly or monthly updates. Features that are seen as not important could also be disabled in offline mode in order to make the amount of offline data as small as possible and keep the complexity of the synchronization down (at least in the beginning).

Editing weekplans
In order to edit the weekplan offline the amount of pictograms have to be limited since all pictograms cannot be stored locally on the phone. An implementation could be saving xx recently/most used pictograms for each citizen.

Take picture as pictogram would have to be limited to a certain amount just like the amount of available pictograms when editing weekplans offline - because it would take too much space on the device if they took 1000 pictures.

A MAJOR issue is if offline changes for the same e.g. activity is made on two different devices - which of the changes should be saved in the online database, when they both come online.
The main problem is deciding what changes should be saved and what changes should be discarded.
Solutions: First write wins, last write wins, not allow the same account offline on multiple devices or other solutions. Evt. discuss with the customer what would fit the Giraf best?

To accommodate this there might need to add more attributes in the offline and online databases in order to deal with and keep track of the synchronization. Examples could be “last_updated_on”, “created_on”, “deleted_on”, “edited_offline” which are timestamps used to see if data should be synched or not and the “edited_offline” could be a boolean.