The Architecture

Version 0.1

Aspect Zero

Terminology

1. An *event* is a specific meeting / appointment / TODO with a specific deadline.
2. A *deadline* is the scheduled time for the above.
3. The *weight* is a static integral property of an event set by the user.
4. The *importance* of an event is a dynamic (with respect to time) property which dictates how the event is handled by the system to some degree.

Aspect One

From the eyes of the end user.

1. **An Event**
   1. This event is the same as the *event* defined in the terminology section.
   2. Every event has an associated deadline and an associated estimated weight. This weight is specified by the user.
   3. Based on the nearness of the event's deadline and the weight we can compute the relative importance of two events.
   4. The relative importance of two events can calculated by a function somewhat like:

      ```
      boolean isAMoreImportantThanB (A, B)
         if ((B.DeadLine – A.DeadLine) > SOME_FACTOR)
            return TRUE
         if ((A.DeadLine – B.DeadLine) > SOME_FACTOR)
            return FALSE
         return (A.Weight > B.Weight);
      ```

   5. Am not looking at these feature as things that will be implemented in this initial version but how about . . .
      1. Optionally attaching a small audio file with an event? Then adding an event would involve two taps and a small oral note. This oral note would be played back as a part of the reminder.
      2. How about photographic data? Specifically allowing them to snap the picture which composing an event? This feature would be one of those not really necessary but fun features.
2. **The Widget**
   1. A small widget in the main screen will have a ticker displaying the most important event.
   2. The Widget changes color according to how near the event is.

Specifically I think a good option would be to let the color approach red as the deadline approaches. Have a look at Widget.JPG for a crude idea.
3. A short tap on the application displays a new activity which shows the user the details of the event being 'ticker'ed on the widget.
4. A long tap on the application creates and displays a context menu which has options for
    1. Looking at the event details.
    2. Postponing / removing the event.
    3. Creating a new event (by default for the same day).
    4. Viewing the calendar.

3. The Calendar
    1. This activity displays a basic calendar (probably a new custom component will have to be built to support the required functionality).
    2. The calendar has three views – DAY, WEEK and MONTH.
        1. DAY would display the events scheduled for a particular day (not necessarily the current) in a list format. The user may delete / modify individual event or add new ones. A possible layout has been given in DAY.JPEG. Note how the first element is light green because the event has already been taken care of. The third one is red as it is the most important event of the day. A better, more appealing of denoting the status of the events would be using icons.
        2. WEEK would show the events of any consecutive seven days in a tabular format. Since we are constrained in display space it would probably be better to show the most important events of each day scrolling in a vertical ticker. See WEEK.JPG for a crude idea.
        3. MONTH would allow for displaying a month of events at a time. I guess we can keep this view similar to the calendar we are used to in our computers. We could set the color of each cell according to the relative importance of the day it represents.
    3. The TODO List
        1. I think it will be nice if the solution also includes a simple, TODO list – the ability to marks things to be done and later checking them off.
    4. Alerting the user
        1. If the user keeps the widget visible then the widget itself may be one means of attracting the user's attention. However we cannot count on it since the user may disable the widget altogether.
        2. A notification appears in the status bar some time before every event. Once the deadline is reached the user is alerted and optionally an alarm is sounded.

Aspect Two

The internal architecture.

1. We will need
    1. Three layouts for the calendar.
    2. One Service in charge of deciding on when to

    1. Change what is being displayed by the widget.
    2. Alerting the user whenever required.
2. We do not have much choice beyond choosing SQLite as the DBMS.
3. Later we might even allow adding / reading of the scheduled events by other applications.

The (plausible) database schema

_ID	DAY	MONTH	YEAR	TIME	WEIGHT	TITLE	BODY	RESERVED
INT	INT	INT	INT	INT	INT	VARCHAR	VARCHAR	INT

The database schema being used now is

_ID	UNIX TIME	WEIGHT	TITLE	BODY	RESERVED
INT	LONG INT	INT	VARCHAR	VARCHAR	INT