Offline Capability of Vaadin Touchkit & mGWT

Due to the fact that no doctor will have the patience to wait for a better Wifi connection and the fact that nobody will be able to set up an 100% Wifi coverage in an hospital, offline capability of medical Apps is mandatory.

So it's clear that the offline capability is quite an interesting point in the evaluation of the both available Frameworks Vaadin & Touchkit.

Vaadin offline Capability:

To estimate the offline Capability of the Vaadin Framework, I searched their website for more information. By looking at the following links, one may notice that creating an offline mode for an Vaadin-App is quite an issue (and not so built in) like in mGWT.
Also their core idea makes clear, that offline modes are not in the focus within development, instead the app logic should run on the server.

However, it is possible to provide offline functionality within Vaadin. This is made by using the HTML5 cache manifest. On vaadin.com/blog/... it says:

“[...]offline web apps are nowadays possible [...] Not easy, but possible. HTML5 contains a concept called cache manifest. [...] The cache manifest hints the browser about files which are used by the web app and which should be strongly cached. Cached web apps can then be run purely from the cached resources if the server cannot be reached. It also helps to load web apps faster on subsequent visits.”[1]

However, within the same blog they advise as follows:

“If your whole application really needs to run offline - don't use Vaadin. Go with GWT or pure Javascript based web app or maybe even with native apps for each of your targeted devices. You'll lose goodies of server centric Vaadin approach like direct access to shared data, but you gain offline support and direct access to hairy browser environment.”[1]

Later on, this is specified in detail:

“If you can meet your requirements by only writing parts of the UI for offline usage, you can have the best of both worlds with Vaadin TouchKit 2.1. The new version automatically writes the HTML5 cache manifest during widgetset compilation.”[1]

But again declares afterwards that making “offline apps” is not the core idea while using Vaadin:

“So why not make every part of you app for offline? The gotcha is that from this point on you are in the browser development world from which Vaadin really tries to save you from. You’ll lose all know goodies of Vaadin and server centric prorgramming model. For example accessing you data becomes harder, your code is not run by an optimized JVM but insecurely by the client and you cannot use most of those common Java libraries that you have used to work with.”[1]

Conclusion on Vaadin:

Some (or even nearly full) offline capability is made possible by using the HTML5 cache Manifest. However, the core idea of Vaadin is to develop server sided apps (and use their advantages), and not providing offline apps.

To see how Vaadin offline mode could look like, one may look at their “Offline Mode in Vornitologist”[2,3] example, where *“Observations will be persisted into devices memory using HTML5 local storage until the server can be reached again. When the real Vaadin part starts again the application suggests to synchronize the locally stored observations to the server so that they can be shared with other Vornitologist users again.”[1]*

[1]
https://vaadin.com/blog/-/blogs/offline-mode-in-vaadin-apps-absurd-joke-or-a-real-solution
[2]
https://vaadin.com/book/vaadin7/-/page/mobile.offline.html
[3]
http://demo.vaadin.com/vornitologist/VAADIN/tutorial/touchkit-tutorial.html

mGWT offline Capability:

The core idea behind mGWT is still a client-side application. So a offline mode is quite not that issue than it is using Vaadin. To keep a high performance, mGWT tries to minimize both, the app-data and the client-server communication. For more detail on the used concepts and the core idea behind mGWT, one may look at [4, 5, 6].

The first big difference is the handling of the already mentioned HTML5 cache manifest. Using vaadin, you might use it, using mGWT you are forced to use it (even without noticing):

“For going offline there is an automatically generated HTML5 offline manifest.”[5]

As mentioned, this is done due to performance issues:

“To make sure your app runs efficiently mgwt does some very cool things:
[…]
Using Caching that works. GWT does some great things to ensure that you only download the script if your local copy is out of date. So if a phone already has visited your page and has the current app in the cache it won`t download the file again and immediately start the app.“[5]

Conclusion on mGWT:

Due to the fact that GWT compiles the application in a way to be as best performing as possible (and in that matter tries to minimizes app data as well as server-client-communication), mGWT offers full offline capability.

[4]
https://www.youtube.com/watch?v=0V0CdhMFiao
[5]
http://blog.daniel-kurka.de/2012/07/mgwt-going-mobile-with-gwt-phonegap.html
[6]
http://code.google.com/p/mgwt/wiki/Why

Overall conclusion

Depending on the app and on how strict the offline capability needs to be fullfilled, one may choose Vaadin or mGWT. In the case of medical environment apps, I would suggest using mGWT.