Developing for Africa - the challenges of building apps for the African environment

Jonathan Haddock
Code Harbour - 2nd October 2019

https://blog.jonsdocs.org.uk
@joncojonathan

I've been developing applications since I was about 16, generally focussed on web applications. I tend to develop in PHP but used to dabble in Microsoft Quick BASIC (years ago) and wrote a proof of concept Android app for my MSc some years ago.

Since working on the eVitabu project with a local charity I've increased the amount of Android / Java development I do. eVitabu is the first tool I've written that has more than 10 users.

The problem

- APF is a Christian charity focussed on transforming lives in Africa
- Training material was circulated on paper
  - Farming
  - Health
  - Christian studies
  - Financial suggestions

While the charity has a strong Christian background and ethos, it doesn't only circulate Christian or Biblical teaching materials although Christianity may run through the materials as a theme.

You can read more about African Pastors Fellowship at https://www.africanpastors.org/ .

The problem

- Paper is expensive to print and ship
- Updates don't happen - once printed there's no change
- Economies of scale meant many copies of documents
  - Sometimes in very obscure languages
- Intra-Africa distribution difficult

There are still many printed documents in storage in Africa. Some are likely no longer relevant or would benefit from being updated.

Distribution networks inside Africa can be expensive, with smaller work being done by bike or by hiring motorbike based couriers. Having to deliver paper documents to individual partners is costly and time inefficient given the distances involved. The other option is that partners travel to a centralised depot to collect resources - even more time inefficient and many do not have transport or the funds to procure it.

APF's partners are based all over sub-Saharan Africa, with contacts in Uganda, Kenya, Malawi, Rwanda and elsewhere. Getting everyone together does not happen often, and it was a real privilege to be at the launch conference with so many partners, from so many places, in March 2018.

eVitabu - the solution

- An Android app
  - Android v5+, with v4 possible
- Easy to use interface
- Searchable
  - Lead to scale challenges
- Resources available offline
- Requires registration & vetting

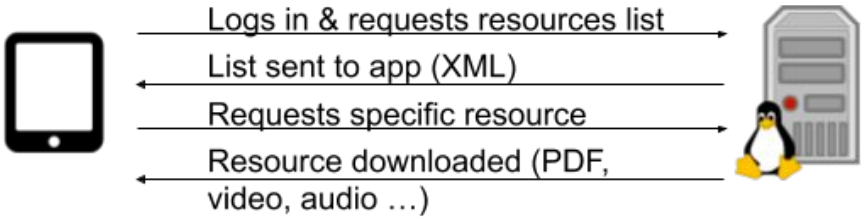
Android version distribution is a challenge. At the launch conference we were providing the hardware, so could concentrate on Android 7 only. Since then we've had prospective users contact us about using older devices, so we now support Android 5. The app will run on Android 4 but that would require the security of the central server to be reduced by enabling TLSv1.0 support. We're looking at Android usage statistics for Africa to determine the best course of action - devices aren't changed out every couple of years like they are in the West.

Users register for access to materials and get vetted by APF staff to ensure they're from known partners. A lot of the contributors are providing materials at discounted rates (or free) on the agreement resources are only accessed by approved individuals.

A lot of time will be spent away from an Internet connection so resources get downloaded to the tablet or phone. During the launch conference (where we provided a free Internet connection) some partners downloaded so many resources they didn't reconnect to our server for over 3 months.

More details about eVitabu are here: https://www.africanpastors.org/evitabu/ .

eVitabu - overview



Hurdles - connectivity: speed, cost & reliability

- Transfer rates can be *sloooow*
- Data is paid for by the MB
  - Same over here, but we have more disposable funds
- Connections may not be stable, leads to repeated download attempts and increased costs
- Necessary to make downloaded data as small as possible
  - A valid goal in any application, but more prominent here

The first version of the app's APK was around 3MB but that later grew to 19MB for reasons that I couldn't determine. This increase was a significant problem, particularly on a slow connection (I simulated this at home on a connection limited to less than 10Mbps - it was painful watching the Play store upgrade the app). Fortunately that's now solved by use of Android App Bundles (https://developer.android.com/platform/technology/app-bundle).

Connection reliability can lead to downloads corrupting or not completing at all. One kick off meeting was plagued with this problem, with a number of reports of missing resources - they all existed in the back end.

Hurdles - allowing for scale

- Early versions of the app sent *lots* of requests
- Initially no caching
- Learning point: load test systems with realistic data sets
  - Test systems of only 20 records aren't the same!
- Initially no compression
  - We're used to browsers automatically enabling Gzip compression - your own app doesn't
- Searching process was inefficient

Originally, each time the user changed activity in the app there was a new connection to the server and a new download of XML. This was fine in early testing, with only a handful of resources, but a real pain after we'd imported over two thousand. Add to that the fact the XML being downloaded was not compressed and we had a runaway problem that would be very costly for partners.

Changes were made to cache the XML response on the device for 5 minutes. This is long enough to browse and select resources for download and the list of available resources doesn't change that regularly - this cache time could be increased. XML data transfers were also compressed with Gzip, massively reducing download times and sizes for the library's metadata.

Searching originally downloaded a new copy of the XML every time a letter was entered into the text box. That was a bad decision and hugely inefficient, something we only noticed after bulk-loading a set of resources. The search now uses the cached data and only begins searching after the third character has been entered.

I wrote a blog post, *Allowing for scale*, which goes into this in greater depth: https://blog.jonsdocs.org.uk/2018/09/12/allowing-for-scale/ .

Hurdles - contextualisation

- I and my co-developers are *mzungu* - white people
- Developing from a Western context
  - Exposure to other apps and GUIs
- Difference in language
  - English is the common language, but simplified
- Icons & symbols may have different meanings

Wherever possible we're looking to contextualise the interface so it makes more sense in Africa. Feedback from our users is always valued and we know there's some areas that need to be adjusted to make the whole experience smoother.

English is used in a simplified form: no contractions ("cannot" rather than "can't") and sentences are kept short.

If an icon isn't immediately understood it's important that it doesn't cause offence.

Hurdles - supporting users

- It's a long distance relationship!
- Email and WhatsApp used a lot
- Local partners assist at set up conferences
  - Sometimes they take short cuts
- Changes to the back end (e.g. the XML schema) must be considered
  - Older versions of the app could be used for many years
  - Backward compatibility

Support was reasonably easy during the launch conference - I was physically with the users along with a team of others that knew how to use eVitabu. Since that first conference there's been at least two more lead by the CEO - when I've not been there. This brings challenges in supporting both the CEO and the end users at the conference, not always achievable during the working day.

Shortcuts sometimes taken by local partners can lead to problems. For example, one group decided they'd share the app via Bluetooth to save having to download the app from the Play store. This caused issues as they were using an old version of the APK, lacking performance and stability enhancements.

We planned the XML schema carefully to make sure it contained all the elements we thought eVitabu would need for some time. The concern is that a partner could be using a very old version of the app to connect to our back end infrastructure. While we'd love them to upgrade, and we encourage them to update, the important thing is that partners can download resources. Additionally, we've built the app to be tolerant to schema changes - additional fields should be ignored if the app doesn't know about them.

Any questions?

Links and acknowledgements

- African Pastors Fellowship
  https://africanpastors.org
- My blog
  https://blog.jonsdocs.org.uk

I'd like to thank Mike, my original co-developer on this project. Adam, my long time coding partner, has also contributed hours to the project.

I'd also like to thank APF's trustees for backing eVitabu, along with our users for helping it grow.

Links and acknowledgements

https://commons.wikimedia.org/wiki/File:Tablet_font_awesome.svg

https://www.needpix.com/photo/100665/computer-server-workstation-pc-hardware-linux-tux