Safety checks

# Workflow approaches

## 1. QR codes – manual scanning

 This is the most basic approach using QR code technology. It will also be the slowest of my recommendations. The camera would be activated and deactivated every time a scan needed to be performed, after pushing a button such as “**scan next**”. The camera’s scanning area would be full screen, which might make it a little easier to center the tag in the image but will slow down the user interface.

The best technology to get this app done in a reasonable timeframe is a hybrid approach using PhoneGap.

The process used with this approach would be something like this:

* Open the app
  + The app will check if you have an active session. Depending on business requirements, we may continue where you left off, or if a certain amount of time has passed we may need to terminate the previous session.
* Provide your login and password and press **login** button
* Press the **begin** **scanning** button.
  + The camera will be activated in full-screen.
*  Scan a location tag
  + If successful, the detail screen will appear to confirm the location scan and allow comments entry
* Enter a comment (optional)
* Press the **back/forward** buttons or swipe left and right to view previous scans and edit their comments (optional).
* Press the **scan next** button.
* Repeat the last four steps until finished and then press **finish scanning**.
  + The app will terminate the scanning session and close.

## 2. QR codes – continuous scanning

This approach also uses QR code technology but is more complex. It should be a little faster and easier to use because the camera would remain active and ready to scan. However, I don’t know what he battery impact will be - it could be significant. The camera’s scanning area would be half of the screen and always visible, which will limit the available screen space a bit.

This app may be too difficult to approach using PhoneGap, but I have located a code sample that sheds light on the approach needed to accomplish it. It may also be considerably difficult using a fully native architecture. Because of the much smaller barrier to entry, I would still suggest first attempting this app with a hybrid PhoneGap application.

The process used with this approach would be something like this:

* Open the app
  + The app will check if you have an active session. Depending on business requirements, we may continue where you left off, or if a certain amount of time has passed we may need to terminate the previous session.
* Provide your login and password and press **login** button
  + After successful authentication the camera will be activated in split-screen mode
* Scan a location tag
  + If successful, the location details will appear below and allow comments entry
* Enter a comment (optional)
* Press the **menu** button to view a list of previous scans and edit their comments (optional).
* Repeat the last three steps until finished and then scan the **finish round** tag.
  + The app will confirm the scan, then terminate the scanning session and close.

## 3. NFC – detailed scan summary

This approach uses NFC technology and is extremely simple. It will be much faster and simpler to use as well because the device is always ready for scanning and can use used without impacting the screen space. Because the keyboard can always remain open, the user interface will be very basic and easy to design.

The best technology to get this app done in a reasonable timeframe is a hybrid approach using PhoneGap.

The process used with this approach would be something like this:

* Scan a location tag
  + The app will be activated regardless of whether it was open or not. It will check if you have an active session. Depending on business requirements, we may continue where you left off, or if a certain amount of time has passed we may need to terminate the previous session.
  + If you do not have an active session, you will provide your login and password and press **login** button
  + If successful, the location details will appear and allow comments entry
* Enter a comment (optional)
* Press the **menu** button to view a list of previous scans and edit their comments (optional).
* Repeat the last three steps until finished and then press **finish scanning**.
  + The app will terminate the scanning session.
  + The app will clear the calendar reminder for the previous round and create one for the next round.
  + Then the app will close.

## 4. NFC – scan list only

This approach uses NFC technology and is also very simple. It should be the fastest approach and is my favorite. It makes the most optimal use of available screen space.

The best technology to get this app done in a reasonable timeframe is a hybrid approach using PhoneGap.

The process used with this approach would be something like this:

* Scan a location tag
  + The app will be activated regardless of whether it was open or not. It will check if you have an active session. Depending on business requirements, we may continue where you left off, or if a certain amount of time has passed we may need to terminate the previous session.
  + If you do not have an active session, you will provide your login and password and press **login** button
  + If successful, the list of locations will appear with a scan date next to the scanned location – the app will automatically scroll to this location
* Tap to expand the comments section (for any location) and enter a comment (optional)
* Repeat the last two steps until finished and then press **finish scanning**.
  + The app will terminate the scanning session and close.

# Fully native vs PhoneGap hybrid

By a **fully native** application I mean one written in a device’s standard development language and environment. Such applications are limited to a single platform or must be rewritten to port to a different platform.

In the case of Apple devices, this would refer to development in Objective-C using Xcode on an iMac. For Android, this would require development in Java using one of multiple available Java development environments and operating systems.

A **PhoneGap hybrid** application differs primarily in how it is developed. **A hybrid application is also native**, but it is derived through a cross-compilation tool that can build apps for multiple platforms from a single codebase. PhoneGap is one of multiple hybrid technologies available. Applications built using PhoneGap are written in HTML and JavaScript.

The inherent difference between these two approaches is defined by the tradeoff between availability of more native language libraries and APIs versus cross-platform development and alternate language support.

## Time to learn

For me, the effort required to learn and to begin developing efficiently with these respective approaches is the most significant factor for how long projects will take. At this point I have scratched the surface of both. I have written basic fully native iOS projects using Xcode as well as a couple simple PhoneGap apps.

However, my heavy familiarity with HTML and JavaScript has helped me **excel in PhoneGap much more rapidly**. The learning required to continue in iOS is a major barrier to entry and will undoubtedly slow me down for some time to come.

## A simplification

There are many comparisons between app development approaches and frameworks online. Ultimately, perspectives vary among widely different perspectives.

[Microsoft has a table on their website comparing these approaches](http://msdn.microsoft.com/en-us/library/jj149679.aspx). It takes a refreshingly simple view and rates then on the following criteria:

* **Investment.** Both the time and money required to build, deploy, and maintain the app must be considered. This includes team salaries, hosting, and maintenance costs.
* **Features.**The features your app needs will play an important role in your decision.
* **Reach.**The number of users you can reach will influence which approach you take.

I think they got their table a bit wrong. They are a bit too friendly to Web and Hybrid applications. A more accurate table would be:

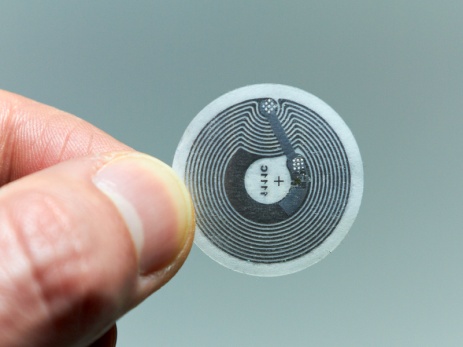
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Investment** | **Reach** | **Features** |
| **Native** | JJ149679.2C432CBE43CCD4519F0DFE784952D7A9(en-us,PandP.10).png | JJ149679.FDD8799F103544A4EE45737E2617C338(en-us,PandP.10).png | JJ149679.13AA606FEE35F142F2D6A5E0FB6BDD87(en-us,PandP.10).png |
| **Web** | JJ149679.13AA606FEE35F142F2D6A5E0FB6BDD87(en-us,PandP.10).png | JJ149679.FDD8799F103544A4EE45737E2617C338(en-us,PandP.10).png | JJ149679.2C432CBE43CCD4519F0DFE784952D7A9(en-us,PandP.10).png |
| **Hybrid** | JJ149679.FDD8799F103544A4EE45737E2617C338(en-us,PandP.10).png | JJ149679.13AA606FEE35F142F2D6A5E0FB6BDD87(en-us,PandP.10).png | JJ149679.FDD8799F103544A4EE45737E2617C338(en-us,PandP.10).png |

In other words, hybrid applications aren’t great… they’re **good enough**. Unless your needs are very specific, it is a good all-around solution.

**Do we need a specific feature** that will require us to pick native for a given project (and give up the benefits of the hybrid approach)? That will only be an issue if the function we need isn’t [already supported](http://phonegap.com/about/feature/) by an API or plugin. In the worst case, however, we could always **develop a plugin ourselves** in the native language.

# https://upload.wikimedia.org/wikipedia/commons/thumb/9/9b/Wikipedia_mobile_en.svg/220px-Wikipedia_mobile_en.svg.pngQR vs NFC

**QR codes** are two-dimensional barcodes which are optimized for scanning on devices with conventional camera lenses. They can store a variety of data including URLs and plain text. However as the amount of data increases, such as in medium-to-long URLs, the complexity increases which effects the scan time and/or accuracy.

**NFC** stands for near-field-communication. It is a standard which uses RFID technology in ca device’s battery to quickly read-and-write data to supported “NFC tags”. NFC tags can also store a variety of data including URLs and plain text. The amount of data does not affect scan time to the degree as in QR codes. Scanning is generally very fast and can store as much data or more than QR codes.

## Summary

This table is meant to summarize some of the comparisons between QR codes and NFC tags for our applications. My conclusion is that NFC provides us with several advantages which make it the preferred technology:

| **Use Case** | **QR Codes** | **NFC Tags** |
| --- | --- | --- |
| Device Selection | Most devices with a camera. Includes **Apple devices**. Low-cost devices probably less effective. Suggested: **iPhone 5C** (**$199+data**) | Limited Android, Windows Phone, or Blackberry devices. No Apple support but peripherals exist. Low-cost devices and tablets more effective.  Suggested: **Nexus 7** (**$199** or less) |
| Overall Cost (tie) | Cheaper to produce tags than NFC. Even in production-ready formats. More dependence on third-parties. Much more difficult to keep spares. Ideal devices undeniably more expensive (camera hardware plus possible data contract). | Low-cost hardware compatibility keeps total costs low. Spare tags remain unwritten until needed, which reduces overall numbers and costs. Prototype tags around 50 cents to $1 each. Heavy duty tags vary from mid to high price ($2.50 to $5 each). |
| Lighting Conditions | Preferably well lit - camera flash probably won’t work. | Any |
| Interaction | **2.5 – 10 seconds** Any distance (of sufficient size that the camera can focus on). Slow and more difficult. Open camera / QR scanner, center image, and wait. Dependant on camera quality. | **0.5 – 2 seconds** Often imperceptibly instantaneous. Easier to position. Must be close proximity - under 10 cm - preferably touching (limits options but provides implicit security). |
| Damage (tie) | Error correction may allow damaged code to be scanned, though scan performance is affected. | Unsure. Tags can be bent and deformed, but once the wires are damaged, reading can become impossible. Heavy-duty form factors should alleviate this concern. |
| Tamper Resistance | Difficult to protect. Heavy-duty printing protects from tears but ink pen could easily affect tag integrity. Easy to spoof through photo copying. | Most tags allow write-protection. Heavy-duty tags protect from damage and allow easy and secure mounting. Spoofing much more difficult. Could be encrypted to protect further. |
| Development | Very easy to prototype codes (just print them out). More community penetration than NFC. Difficult to develop for camera, stabilize, and optimize performance. APIs are less simple and best ones are proprietary. | Simpler development model is more dependable and much easier to code. Prototyping still easy but requires purchasing tags. |

## Ease of use

A major concern of mine has been the speed and convenience of the technology we choose. I tested the differences and found **NFC tags were significantly faster and easier to use**.

**QR codes** require use of the camera in proper lighting, which involves viewing the barcode on-screen, focusing, and centering it in the viewport. In my tests, the scan rate ranges from **2.5 seconds to 10 seconds**. In fact, the minimum 2.5 second scan is a best-case result and is not realistic. In that test, the device was perfectly centered on the barcode when I launched the camera and in perfect lighting conditions (reading form an LCD screen). Also, the camera had been launched several times recently which probably improved performance. QR code performance alongside a mobile phone LED flash is notoriously bad and unusable, resulting in blown-out images which can’t be scanned. So, all tags must be in locations with sufficient external lighting. Otherwise, scanning performance will be significantly affected or may not function at all.

**NFC tags** only require touching the battery of the device to the tag. The relative simplicity (no camera and easy placement) made this much easier to use. The scan speeds for different NFC tags are publicly rated, and the ones I would use are rated for under 0.5 seconds scan speeds. In my test, the scan rate ranged from **immediate to 2 seconds**. Most scan delays seem to be attributable to not covering the entire tag with the battery. If the battery location is not obvious, this could be remedied with a Sharpie or sticker to mark the device.

## Devices

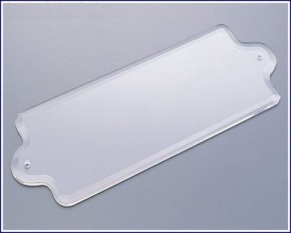
A notable benefit to **QR codes** is they can be scanned by existing Apple phones or by an iPod Touch since it has a camera. However, if targeting low-cost devices, Apple hardware may not perform well enough to justify using QR codes. The current generation of iPod Touch camera is roughly 2.5 years behind the iPhone and other competitive hardware. I have found reports of [difficulty scanning QR codes well on the current generation of iPod Touch](http://forums.adobe.com/thread/1017426). While I expect it will work. It will probably be noticeably slower and more difficult to develop.

Less devices support **NFC**, however that number is growing and includes multiple low-cost devices such as tablets - which won’t require a data plan. Unfortunately, Apple does not support NFC. However, several other manufacturers do, including phones and tablets using Android, Windows Phone, and Blackberry operating systems. [Multiple](http://www.gizmag.com/flojack-nfc-reader-writer-ios-devices/24891/) Apple [device](http://www.icarte.ca/) [peripherals](http://www.gizmag.com/flojack-nfc-reader-writer-ios-devices/24891/) are also available for iPhones and the iPod Touch. One sells for $60 and has an [open-source SDK](https://github.com/flomio/flojack_ios) so there is an option for the agency’s current iOS devices.

The Nexus 7 is a small tablet (smaller than the iPad mini). It has NFC and goes for **$199** (or cheaper for the 2012 model). An added benefit to a tablet is a larger battery which will increase the surface area for scanning and the device lifetime.

## Confidex NFC Tag Ironside Micro NTAG203 (NFC Forum Type 2)File:QR Code Damaged.jpgProduction use

When installed in a correctional facility, we must consider certain factors to evaluate a product’s viability. These could include: durability, tamper resistance, ease of production and replacement, and cost of duplication.

**QR codes** are cheap to print for testing, but require coordination with third parties or purchasing expensive printing equipment to produce production-ready labels. We would obtain heavy duty labels for tear and removal resistance, which vary in cost. It may be necessary to cover labels with clear (i.e. acrylic) door plates if tampering is a serious concern. Still, barcodes cannot be well protected from overwriting with pens or sharpies to damage or even change the embedded code.

Because each barcode is a specific code, we could not keep spares without keeping a spare of **every barcode**. Printing a set of many different barcodes will also cost more than reprinting a single barcode (as companies would do in an advertising campaign).

**NFC tags** cannot be produced in-house so they need to be purchased. This raises costs for prototyping ($0.50 to $1 for prototype tags). Cost differences for production tags are more competitive but probably still higher ([$2.50](http://www.buynfctags.com/heavy-duty-anti-metal-nfc-token-1k.html) to [$5](http://www.nfcnetstore.com/Confidex+NFC+Tag+Ironside+Micro+NTAG203+(NFC+Forum+Type+2)/p/113/) each depending on type and quantity). Because NFC tags are written to digitally, we would need to encode them ourselves. However, we could keep blank spares and would not require new orders every time a tag needed to be replaced. An app could even allow on-site staff to encode and write-protect new tags themselves. This would end up saving time and remove dependencies on third parties. [Heavy duty tags](http://www.nfcnetstore.com/Confidex+NFC+Tag+Ironside+Micro+NTAG203+(NFC+Forum+Type+2)/p/113/) are also much more [robust](http://www.confidex.com/ticketing/images/nfc/pdf/Ironside_Micro_NFC_Datasheet.pdf) and should not need frequent replacement. Tampering and spoofing can be made nearly impossible if encrypted.

Overall cost differences are not entirely clear. I suspect that the ease of keeping **spare NFC tags may result in lower costs and less workload during replacements**.

## Development

Because **NFC** scanning uses less moving parts than a barcode scanner, it will be easier to develop successful applications. They require less fine tuning than QR codes. To maximize QR code performance, the character set must be optimized, as should the message length, physical size, colors, and other printing techniques (such as developing vector graphics and coordinating with vendors to develop effective sizes and high-quality output). My proof-of-concept has already shown than NFC device APIs are easy to develop against and will allow for faster development times.