Opal Versioning

The versioning of the app will be handled via two numbers: (1) the version number; and (2) the build number. The version number will be the publicly facing version while the build number will be internal to our builds of the app (every commit to a main opal branch).

For the main version version we use SemVer for versioning. For the versions available, see the tags on this repository.

The main Opal branches are: `prod` for production, `preprod` for pre-production, and `staging` for development.

Version number

The version number will have the following format: `major.minor.patch`. We will decide ahead of time what features the new version of production will contain. If it contains too many new features and views, we may decide to bump up the major number. On the other hand, if it contains fixes to current views, or a re-design of current views we then may simply bump the minor version. If we set the production version to 1.8.0, the `preprod` version will become 1.8.0-rc, while the `staging` version will be 1.8.0-beta. These version numbers will not be updated in `staging` and `preprod` until they are updated in production which will only happen when we have determined the set of features the new release will contain.

Build Numbers

A build number will simply be an integer. e.g. `1817` Build numbers are used to manage internally our versions of the app, i.e. anytime we merge a commit to `staging` or `preprod` a new version of the app will be built with a unique build number.

The `staging` branch

In the normal flow, a developer will develop a feature in a branch off the `staging` branch, once the developer is ready to merge, a merge request will be made to `staging`, once this request is approved, a new version of the `staging` app will be built with the current bumping the current staging *build number* by one. Every developer/manager or related party to `staging` then automatically gets this released.

A commit to `staging` has requirements for specificiations and code reviews, once fulfilled and merged, the developer that committed the request will provide a JIRA task with a set of tests for the given commit, once that task has been fulfilled by another developer, the feature will be considered stable enough to go into `preprod`.

The `preprod` branch

The serves as a testing area for clinical staff involve with the Opal project. The staff will be provided with a list of new features, specifications and a set of high-level tests in terms of functionality. They will make sure the

app fulfills this requirements adequately from a clinical point of view.

Merging from staging

To merge from `staging` to `preprod`:

1. Create a branch from `staging` called `staging-preprod-1.8.0` where the last appended string, `-1.8.10` represents the version number.
2. Make the following changes to the `preprod` environment files under the `./env` directory:
    1. Any change to the config.xml in the staging environment has to be migrated over to the config.xml of the preprod
        - Diff the two files in the environment folder to find the differences.
    2. The buildNumber will be set to 0 (these is represented by two variables `android-versionCode`, and `ios-CFBundleVersion` in the `config.xml` file and `buildNumber` in the `opal.config.js` file.)
    3. Migrate any change to the environment variables for the Opal app set in `staging` (`opal.config.js`).
    4. Set the `preprod` version in the `config.xml` and `opal.config.js` to the proposed version for the master branch albeit with the `rc` label attached at the end indicating the following commit in `preprod` is a release candidate. e.g. `1.8.17-rc`.
3. Make a MR to the `preprod` branch.

Once the MR is merged, the commit should automatically build the `preprod` app, and sending the app to all the `preprod` related parties.

The production branch

This branch will represent what is currently in the App Stores for iOS and Android. In this branch we will purely use the version number (`major.minor.patch`) to control what will be displayed. There are two flows to update this branch:

The normal flow

This goes through the flow of `staging`->`preprod`->`prod`. during this flow, we know exactly what features will be in the next version of `prod`, and thus we know the version that will eventually be released. In this case, depending on the features release we will bump either the minor or major version number.

Hot fixes flow

If there are any hot fixes in production here is the procedure:

1. Create a branch off `prod` named `1.8.18-hot-fix` where the patch number indicates the patch number for the fix.
2. Fix and test the fix of this problem in this branch.
3. Create an MR for another developer to review.
4. Merge the change into production.
5. If possible and desirable bring back (cherry-pick) the commit into the `staging`/`preprod` branches.
   NOTE: Bringing the commit back upon fixes is a mandatory step (if it makes sense)

`preprod` and preparing for production

Follow the steps given in the (Merging From Staging)[#merging-from-staging], in this case however, set the appropriate files to version stipulated by `MAJOR.MINOR.PATCH`.