Ushahidi v3.x, Development process

Hacking on the Ushahidi Platform

We use Phabricator for all project management, issue tracking, and code review. You can find our instance of "phab", as we call it internally, at https://phabricator.ushahidi.com/ After you have signed up for an account, please read the help guide and follow the instructions for setting up `arc`.

The `master` branch is considered "stable" in that the build should be passing and all tests should complete successfully. All development is done locally or through feature branches. All changes must be code reviewed through the Differential tool on phab. Each modification should be a complete unit of change and not break any test. Any new feature should come with a spec test, unit test, or feature test.

Progress Tracking / Workflow

Ushahidi practices the Agile method of development, with a continuously changing master branch and timed releases. New releases are packaged bi-weekly on Thursday afternoons (US/Eastern time). You can check what tasks are currently being worked on by following the project workboard on phab. Any task that is planned, but not yet being worked on, will be in the Backlog column. Tasks in the Next Release column are to be for the next release. Anything that has been shelved for a later date, or has very low priority, will be in the Bike Rack.

Bug Reports and Features

Bug reports should include:

- A short description of the bug.
- The exact steps needed to reproduce the bug.
- What the expected result should be.
- What the actual result is.

Feature specifications are more loosely organized, but should always follow these rules:

- All specs will be defined as a task (or tasks) on phab.
- All specs will have their dependent tasks properly selected and visible.
- All specs should list any special requirements or resources needed to complete the task.
- All specs should include a general idea of what the implementation will be, or how it would be presented in the UI.
- All specs may also include special considerations.

Testing Changes

We prefer that tests are written before writing code, but we do not strictly enforce TDD. However, we do require that:

- Every new feature be tested.
- Every change in functionality have a regression test.
- All business code (the stuff in `src/`) be spec'ed using phpspec.

For testing, we use a mixture of phpspec, phpunit, and Behat for functional tests. We do not currently test the client-side JS code, but we plan to. (Recommendations on this are greatly appreciated!) See 3.x Testing for info on running the tests.