Mobile Hybrid (Ionic / Heroku) Buildpack

Goals

The primary goals of this project are:

1. Create a build pack for hybrid mobile apps using Ionic (other build packs may address other methods or stacks for mobile delivery).
2. Create the plan for ongoing maintenance and use of the buildpack to accelerate mobile app build in the community

When complete the build pack should allow our clients to "bootstrap" mobile application delivery where a hybrid solution is desired.

General Description

The buildpack will consist of a demo Ionic application showcasing simple examples of all of the most common mobile features. "Under the hood", the application code will be structured in a highly modular fashion. Separate code modules for each features will be created so that applications based on this buildpack can "pick and choose" which features to implement and customize from this buildpack.

In addition to this front-end demo application, the buildpack will provide a backend node application, also demonstrating the most common features used by mobile applications. Various front end features will require this capability as well (such as push notifications and email services). This version of the buildpack will use Node on Heroku for the backend - however the API will be designed to be generic so that different backends can be swapped in. (e.g. the front-end developers will not need to know the details on the backend).

In combination these two pieces will provide a solid and consistent starting point for many different types of hybrid mobile application builds. It will both reduce cost and greatly accelerate time to delivery. In addition, as the community's expertise with the buildpack increases, time to delivery of the final application will further accelerate.

Features

- Authentication
  - Oauth authentication
  - Direct user authentication (DB based)

- Social Features
    - Twitter
    - Facebook
    - [Let's keep it to two for now, can expand later]
- Analytics
    - Google Analytics
    - [Would like two, what's the next most common, free one?] (Maybe KissMetrics or MixPanel?) I think they both have
- User roles
    - Roll based API call authentication
    - Roll based application features
- Database
    - Relational (Postgres is sufficient, it's easily translatable to pretty much any other relational DB)
    - Document (NOSQL - Mongo)
    - Generic CRUD for both relational and document
- Thirdparty API access (through the backend)
    - Demonstrate access to 3rd party APIs through the back end
- Mobile Hardware Access
    - Camera access (Photo and video demonstration)
    - Fingerprint (maybe, maybe "phase 2" since it's so non-standard)
- Email capability (from the app and via the backend)

Front End

The front-end result of this project will be a "kitchen-sink" style demo application with a menu to allow the user to view and test each of the different features described above. Although this will be the resulting **view** of the application, what's really important is what's under the covers - modular, reusable pieces that can be used in a client application build. During development each unneeded piece can be disabled / removed and development and customization can take place on the technologies that will actually be used.

Back End

The backend will serve two purposes. Like the front-end, it will provide a modular starting point to bootstrap development of new applications. This will include modules in support of all the features above. In addition, it will provides specific services to enable the demo application.

Other Notes

- All code will be developed in the crowd and all IP will be owned by Topcoder. This will allow us to use buildpacks as a starting point for any project with any client.
- Future / Second steps may be to rebuild the backend for Amazon and Bluemix.
- We will run all API calls through an API gateway (Apigee).
- All front-end and back-end code will have full CI management.
- Potential development or adoption of coding standards for buildpack development (Rollbar or similar).
- Potential use of Feature Management (such a LaunchDarkly).
- Also see HashiCorp (Consul and Terraform)

Initial Creation

The initial creation of the buildpack will take place over the course of several challenges. Due to the modular nature of the project - the we will be using small iterative challenges to create and integrate each piece.

Challenge List [Still to be finalized]:

- Documented API (swagger) for all features described above
    - Mock Interfaces for all calls
- Basic modular, git tracked, Heroku deployable Node backed with the above APIs stubbed out
- Implementation of API services (specific challenge list TBD)
- Ionic front end initial development with API calls stubbed (no design - standard controls)
- Ionic front end features implemented (specific challenges TBD - likely 2-3 challenges)
- Bug testing / fixing

Plan for Maintenance

Major Maintenance

Every 6 months, a technology review will take place on each feature present in the buildpack. Followup challenges will be run to replace any outdated features with their modern equivalents and update all software versions to current stable releases. Bug hunts and fixes will be run again.

Ongoing Maintenance

Smaller fixes and changes, based on experiences "from the field" that need to be made to the buildpack will be managed as need by the buildpack owner (possibly utilizing the community).