**Для дебага, дополнительных тестов и продакшана:** из папки с проектом все доп инструменты оставлены и загружены. При компиляции в продакшн ионик не компилирует инструменты разработки, что уменьшит вес приложения в разы. Прежде всего в новом терминале надо перейти в директорию ServerAPI и запустить скрипт **npm run server**. Для эмуляции работы приложения в браузере: в директории с проектом (Admin или App) пишем в терминале **ionic serve -l**. Не забывайте флаг -l, потому что иначе приложение будет эмулироваться под платформу браузера. Spec тесты я не проводил, т.к. просто не умею это делать. При сборке готового приложения не забудьте прописать флаг --prod для компиляции проекта в продакшн. Все приложения написаны на дефолтных компонентах ионика из коробки, так что с кросс-платформенностью проблем быть не должно.

**Архитектура приложений:** Сам API написан с концепциями model controller, я постарался сделать максимально понятную архитектуру. API использует облачную базу данных mongo, кластер привязан к моему гугл-аккаунту. Для доступа к кластеру через mongoDB compass в файле MongoCluster лежит url для коннекта. Этот же url использует и наш API. Через кластер можно будет посмотреть всю бд изнутри. Спойлер: там лежит только несколько коллекций с данными. Если захотите сменить базу данных на другую, в папке config лежит файл keys.js, там и записана ссылка на кластер. Возможно придётся переписывать API, так как он заточен конкретно под монго. Url естественно localhost, при выпуске в продакшн у самого API и в исходном коде приложений его нужно будет менять. В админ-приложении вся работа опирается на сервисы, каждый из которых запрашивает у API данные и обрабатывает их. Все сервисы лежат в папке src/app/shared/services/. Также только сервисы делают все http-запросы к нашему API. Сервис dataTransfer можно игнорировать, т.к. он отвечает только за обмен данными в компонентах и их обработке. Во всех остальных сервисах есть переменная **apiUrl**, в которой необходимо будет сменить ссылку на наш API, когда его домен будет зарегистрирован. Аналогичные действия надо будет совершить и в приложении-клиенте. Сервис userData также можно игнорировать, т.к. он отвечает только за сохранение выбора пользователя. Иконку приложения можно добавить в папке src/app/assets/icon/favicon.png. TS компилируется в стандарт ES6.

**Дополнительно:** при эмуляции в браузере ответы от API приходили в формате httpErrorResponse, что очень сильно резало мне глаза. На работе приложений оно не сказывается, но некрасиво. Как пофиксить я не знаю, но просто говорю что такая проблема есть. Также на кроссдоменные запросы ругался CORS, я это пофиксил в API прописав app.use(cors()); Всё что связано с запросами лучше тоже ещё раз проверить. На компе при разработке, по крайней мере, всё работает нормально.

Если есть ещё какие-либо вопросы, тут я тоже оставлю свои контакты:  
@tinkldeath – телеграм,

[dropdeadit@gmail.com](mailto:dropdeadit@gmail.com) – почта.