未熟なチーム開発

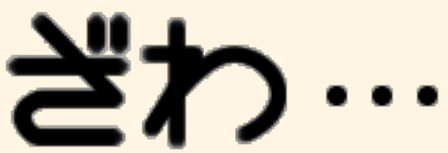
@kgmyshin

こんな状況①

- だいたい20万行弱くらい、画面数は50弱くらいのAndroidアプリを作ることになった
- アニメーションもりもり
- クイズだったり音声認識だったりする機能があって、API叩いて表示するだけじゃないのがそこそこある
- iOSはすでにリリース済み

こんな状況②

- iOSの時は、ベテランiOSエンジニア2人と自分だった
- Androidのメンバーは自分と新卒とiOSエンジニア
- 期間はiOSの時と同じくらい



だったが、結果は

- 遅れることなく無事リリースできた
- 今のところ、問題なく運用できてる

この発表では

- 初期にやってよかったことを発表します
- やらなくてよかったことは時間に収まらない

ので、興味あれば聞いてください

やってよかったこと

- 設計方針をしっかり決めた
- テストは必ず書いてもらった
- Theme, Styleの追加/変更禁止
- colorの追加禁止
- dimens, strings, shape, selectorはこまめに切る
- クラス図書いてもらった(新卒向け)

設計方針をしっかり決めた

例えば、ある機能を作るとなった時に

AActviity, APresenter, AUseCase, ARepository

を作れば良いと誰でも想像できるレベル

ここら辺を守っておけば何でもいいと思う

- 非同期
- 永続化
- ビジネスロジック
- イベント管理
- Viewの管理
- フレームワーク独自のクセの整理

こうすることで

- ゼロベースで考えなくていいので早い
- フレームワーク独自のクセをあらかじめ整理しているのでiOSエンジニアでもAndroid独特の罠にひっかかりづらい

テストは必ず書いてもらった

- テストの内容が微妙でも、書いた方が良い
- 書かない場合と書く場合で、全然コードが変わる
- mockできるコードかどうか
- 後から書くと決めちゃうと、mockできないコードになってることが多くて到底払えないコストになりうる
- 設計方針を細かくしてるので、テスト方針も細かく書いていて、基本的に機能実装時同様そんなに考えることがないので早い(機械的にできる、物や粒度によるけど一つのプルリクにだいたい30分から1時間くらい)

colorの追加禁止

- あらかじめデザイナーにもらってるカラーパットを登録している
- そのため、追加は必要ないはずなので禁止

Theme, Styleの追加/変更禁止

- あらかじめ本当にどこでも使うものは作っておく
- それ以外は基本的に追加禁止
- 乱立されると収拾がつかなくなる
- 勝手に変更されると、意図しないところの見た目が変わってレビューで確認できない

dimens,strings,shape, selectorはこまめに切る

- 本当にどこでも使うものはあらかじめ作っておく
- あらかじめあるものは変更禁止
  - 勝手に変更されると、意図しないところの見た目が変わってレビューで確認できない
  - なので、それ以外は基本的にこまめに追加してもらう

クラス図書いてもらってプルリクの説明に貼ってもらった(新卒向け)

- 導入前
    - 設計方針不理解
    - 我の強いコード(すでにあるコードを参考にしない)
    - メソッド名、クラス名がわかりづらかった
    - レビュー時に発覚して、ほぼやり直しみたいなことがあってすごく時間がもったいなかった

クラス図を描いてもらったら

- 設計方針の食い違いをあらかじめ抽出できた
- メソッド名、クラス名をレビューしやすい
- クラス図を書くことで、それらに集中して考えるので、クラス設計やネーミングの能力が上がりやすい
- 二度手間ということがなくなった

一言にまとめると

- あらかじめ考えられるものは、先に考えきってしまうしておこう
- グローバルなものはあらかじめ作って変更不可として、細かいものは一切継承など考えずに作ってしまった方がよい

とはいえ

- 時間が経つにつれて、みんな成長していくのでどんどん任していこう

- 最初は武器作って敵倒して、その武器を配っていくイメージ

- だんだん武器作りにも参加してもらうようなイメージ