mercari
DevDojo

DevDojo - Mercari Mobile Development (2024/04 ver.)

@klausa (Jan Klausa)

@justin999 (Koichi Sato)

Today’s contents

1 Mobile development cycle overview

2 Details in each phase

3 Tech stack

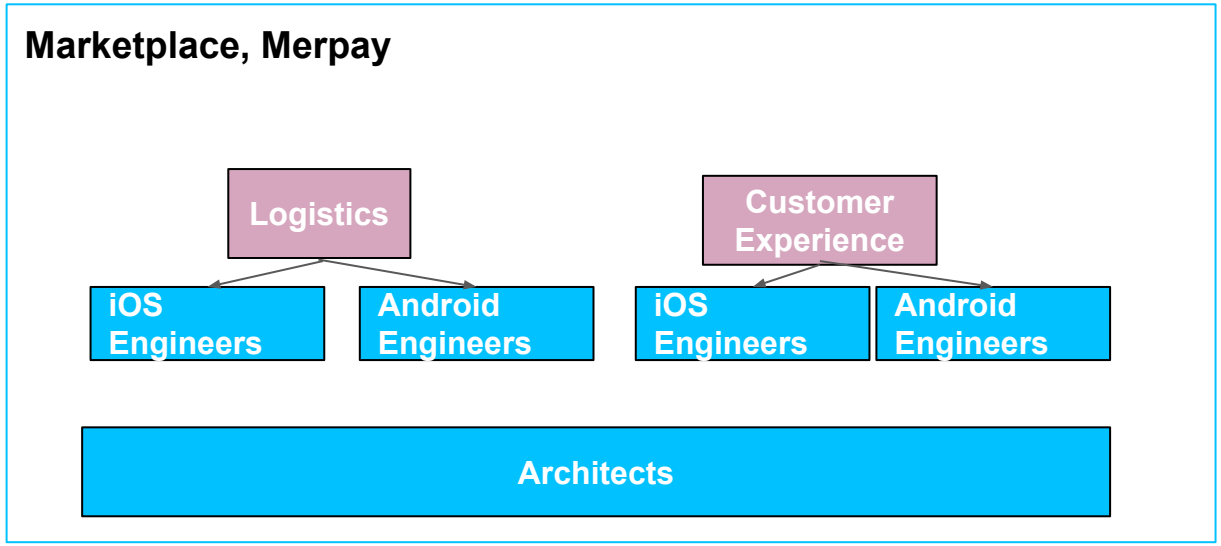
Term definitions in this presentation

- Engineer: Software Engineer
- PM: Product Manager
- QA: Quality Assurance
- PiC: Person in Charge

- JR: Japan Region
- MK: Marketplace
- MC: Mercoin
- MP: Merpay
- SZ: Souzoh (Mercari Shops)

1. Mobile development cycle overview

Team Structure



Development cycle overview

1. Create Specs (PM)
2. Create UI Design (Designer)
3. Write Technical Design Doc (Engineer / PiC)
4. Ticketing & Planning
5. Coding & Create PR(s) (Engineer)
6. Self-QA (Engineer)
7. QA (QA Engineer)
8. Dogfooding
9. Release & Monitoring (Engineer/QA)

2. Details in each phase

(each team is different!)

Create Specs (PM)

- PM creates feature specs
  - Created in Confluence
  - Client Engineer serves as the engineering contact
    - Review the Spec, provide feedback, and raises any concerns

Create UI Design (Designer)

- Designer creates UI design
  - Created on Figma
  - Client Engineer should check about:
    - Transition way (Push/Full-modal/Half-modal)
    - Use of Design System
    - Error states, Empty states
    - Platform specific things (iOS/Android)

Write Technical Design Doc (Engineer)

- Engineer writes Technical Design Doc (TDD) before actual implementation
  - DesignDoc DevDojo
  - Writing Technical Design Doc helps us to understand:
    - what we should do
    - technical design
    - APIs we need to call
    - edge cases
    - conflicts
    - platform specific details

Ticketing & Planning

- Scrum members create tickets and have planning meetings
  - Ticket type hierarchy:
    - Epic: root ticket for a feature
    - Story: User stories ticket for 1 epic ticket
    - Task: fine-grained tickets for 1 story ticket (optional)
  - Have planning meeting based on created Story(Task) tickets
    - We use scrum as development style
    - Each teams controls their own process

Coding & create PR(s) (Engineer)

- Engineer writes code!
  - Flow
    - 1. Engineers code
    - 2. Create PR (Pull Request) on GitHub
    - 3. Get reviews & make modifications
    - 4. Merge after approval
- CI status is important
  - Unit Tests, UI Test, Snapshot Tests, etc...
  - Distribute to employees

Coding & create PR(s) (Engineer)

- We use Trunk-based development style as branch management
  - Keep PR size small, Merge quickly
  - Master branch is always ready for release
  - Protect new features or bug fixes behind Feature Flags to keep master stable

QA (QA Engineer)

- We have dedicated QA members for each team
  - QA = Quality assurance (≠Manual tests)
  - Each team has QA members
  - Mobile Engineers are also responsible for QA
  - TestRail is used for managing test cases

Dogfooding

- Some teams do dogfooding before/after QA as well
  - Flow
    - Create a dogfooding confluence/doc
    - Schedule a meeting with the team to dogfood together
    - Discuss on feedbacks and take actions accordingly

Release & Monitoring (Engineer)

- Release
  - We need to get approval from the platforms to release new versions.
  - Flow
    - Cut release branch (Wednesdays at 4PM)
    - Release judgement test (Thursday)
    - Submit to each app store (App Store, Google Play Store) (Thursday)
    - Waiting for reviews from Apple and Google
    - Release (Next Tuesday)
- Monitoring
  - Engineer should also check crash status after release
  - Hot fix release is needed if crash status is severe

3. Tech stack

Feature Flag & Laplace

- Feature Flag (a.k.a. Remote Configs)
  - Feature testing (≒ A/B test) system controlled by backend config
  - It's used for not only new features, but **also refactoring, etc**
- Laplace
  - New generation log platform in Mercari (Old log platform is Pascal)
  - Batch and Realtime log sending
  - Well-structured log schema

Feature Flag & Laplace

- Feature Flag × Laplace = Data driven development
  - Create a feature with some variants and integrate appropriate log for analyze
  - We can decide which variant is the base based on actual data (not inference)

JSON & Protobuf

- We use JSON and Protobuf for HTTP request and response
  - JSON: used for legacy endpoints (≒ mercari-api); but some teams still use it for new development
  - Protobuf: used for microservice endpoints

- We DON’T use gRPC for client
  - gRPC is used for communications between microservices

Design System

- Design System provides consistent UI Design styles and components
  - There are some layers similar with Atomic Design
    - Styles, Elements, Components, Templates
  - ref: Figma, ZeroHeight

Playbooks

- Used for UI Tests, helps with UI development
- Component library /
- Whole screens
- Standalone
- Separate target

Thank you for listening!

Q&A