JaSST' 26 Hokkaido

Google App Scriptを活用した軽量モック環境による外部連携テストの効率化

東急株式会社
デジタルプラットフォーム URBAN HACKS
朱峰 錦司

自己紹介

URBAN HACKS
TOKYU CORPORATION

■ **朱峰 錦司**(あけみね きんじ)

- 東急株式会社
  デジタルプラットフォーム URBAN HACKS
  - ✓ 本日の発表内容のテスト対象システムのプロダクトマネージャー
  - ✓ QAエンジニアコミュニティリード

- これまでのキャリア
  - ✓ 2009.04–2021.03: 大手SIerにてQA/アジャイル領域にてなんでも屋
  - ✓ 2021.04-2025.06: 検証専門会社にて自社B2B製品群のPdM/PjM

本日の主題

小規模プロダクトチームが
テスト環境のない外部システムに対して
“十分に使えて維持できる”モック環境を
どう設計したか

お品書き

1. 本施策の背景

2. 外部システム連携テストにおける問題

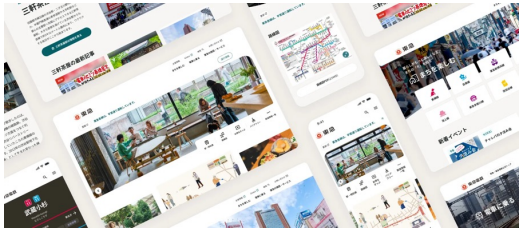
3. モック環境導入のための課題

4. 導入したシステムとその過程

1. 本施策の背景

当組織で開発・運用しているプロダクトの例

URBAN HACKS TOKYU CORPORATION



東急・東急電鉄公式サイト

両社の企業情報のほか、東急線の運行情報や各駅情報などを
よりわかりやすく掲載。また、東急(株)グループ各社が展開する
事業・サービスや、東急線沿線を中心とした「まち」に関する情報も発信します。

Tokyu Plusアプリ

東急カード会員向けの利用明細を確認できるアプリ。
沿線エリアのタイムリーでおトクな情報や、ポイントがお得に貯まる方法なども
紹介しています。

Shibuya TOQ Passアプリ

渋谷の東急株式会社関連施設やテナントで働く方を対象とした、
特典・情報提供アプリです。アプリを店舗で提示すると、対象店舗にて
優待サービスを受けられるほか、渋谷で開催されるイベント情報や、
渋谷をテーマにした記事なども配信しています。

3つのプロダクトの共通点

URBAN HACKS TOKYU CORPORATION

東急グループのイベントが、
どこでも宣伝されている！

東急・東急電鉄公式サイト

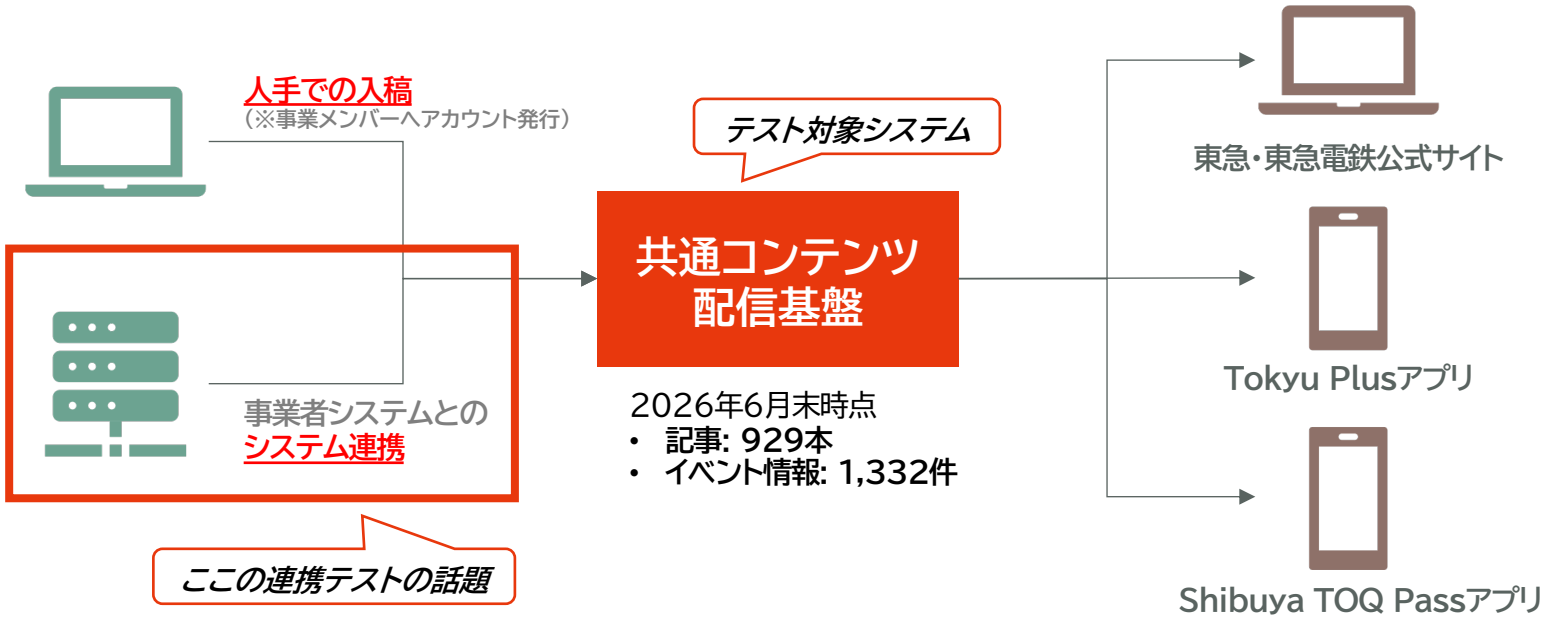
Tokyu Plusアプリ

Shibuya TOQ Passアプリ

共通コンテンツ配信基盤

URBAN HACKS TOKYU CORPORATION

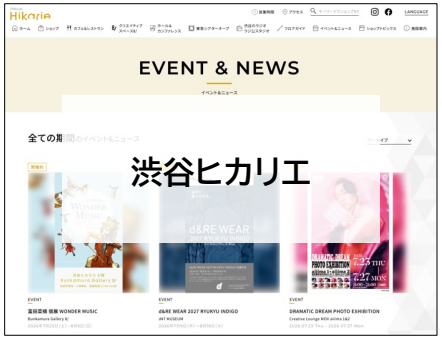
■ 東急グループの事業や主催イベント情報の流通・露出の加速のため、CMS(コンテンツ・マネジメント・システム)基盤を企画・開発・運用しています

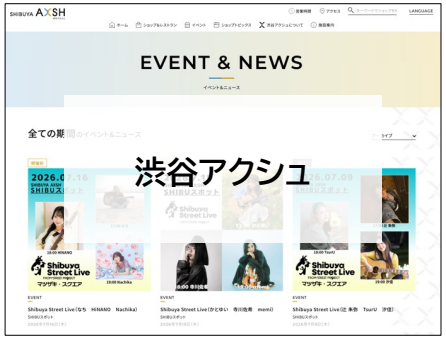


2. 外部システム連携テストにおける問題

連携している外部システムの例

URBAN HACKS TOKYU CORPORATION





外部システムの特徴: パッケージ製品の利用

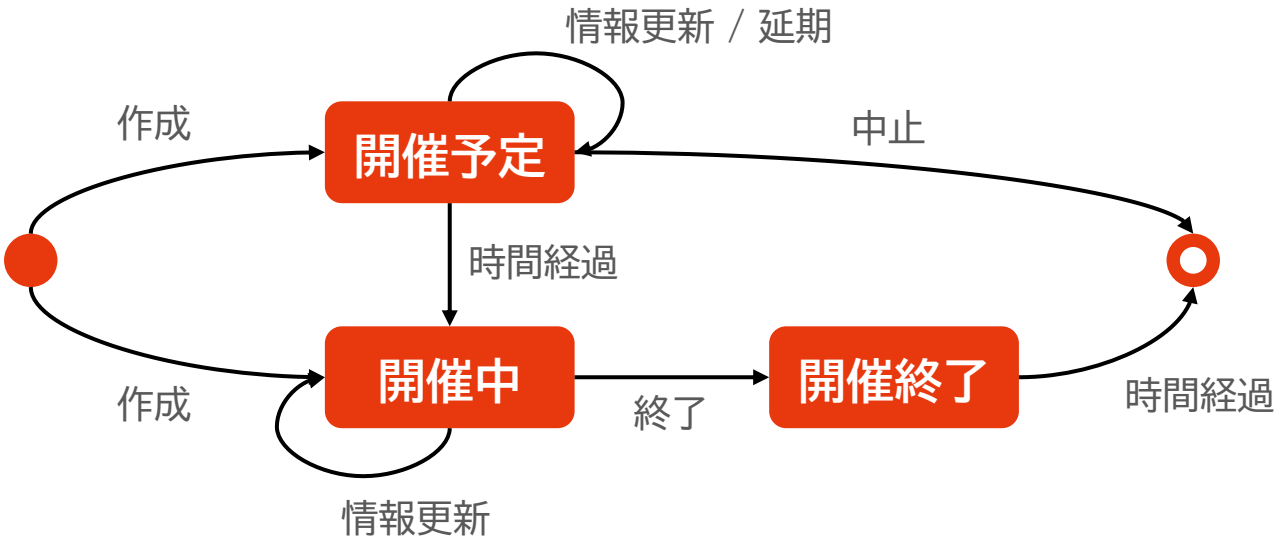
URBAN HACKS TOKYU CORPORATION

■ 既製品(のカスタマイズ)で完結するため、いわゆる本番環境のみ提供され、
自由につかえるテスト環境が存在しない

イベント情報のライフサイクル

URBAN HACKS TOKYU CORPORATION

■ 多様な状態遷移が起こり得るため、情報を同期をするシステムとして、全パターンをしっかりテストする必要がある



どうテストしていたか？

施設担当者との調整コスト

- 許可が取れた施設については、本番環境のアカウントを払い出してもらい、**状態遷移のテストに必要なパターンのデータを、直接本番環境上に作成**してテストを実施

本番サイト上への意図せぬ情報表示リスク

既存の本番データの意図せぬ編集・削除リスク

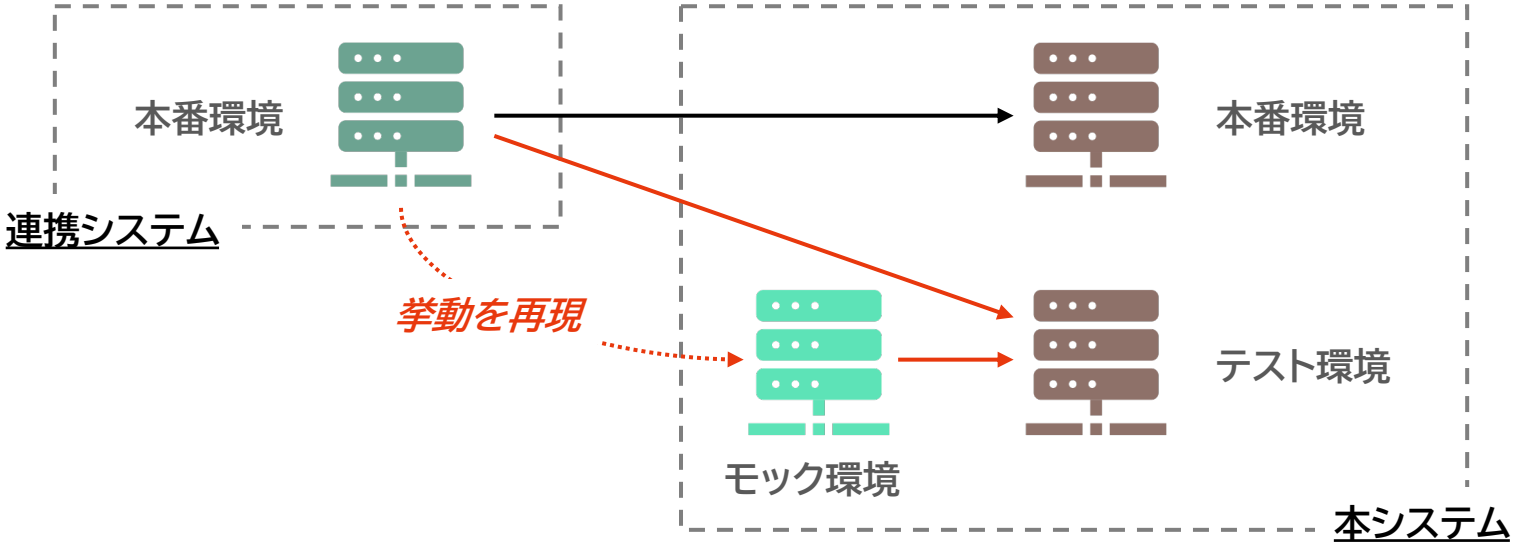
- 許可がとれなかった施設については、既存の本番データのみでテストを行い、**カバーできない遷移パターンは許容**

本番不具合リスク

モック環境の導入

URBAN HACKS TOKYU CORPORATION

■ 先方にテスト環境がなく、かつ、本番環境を使わない、となると、
当方にて**擬似的なテスト環境＝モック環境**を用意するしかない



3. モック環境導入のための課題

本プロダクトの開発体制

■ 非常に小規模なチームである（※本施策当時は）

体制やその他観点を考慮した際の課題

1. 本番環境・モック環境の使い分けの方針立案

2. 構築・維持コストの最小化

3. QAメンバーのみで対応可能なモックデータの差し替えの仕組み

4. 一定のセキュリティ水準の確保

4. 導入したシステムとその過程

現時点のモックシステム構成

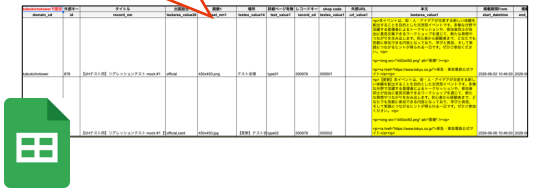
URBAN HACKS TOKYU CORPORATION

モック環境

*Google Spreadsheet*で
モックデータ管理

*Google App Script*で
連携システムと同じ
データフォーマットに変換

*Google Drive*上のファイルを
*Google App Script*で
Web APIとして公開



JSON
ファイル

本システムのテスト環境

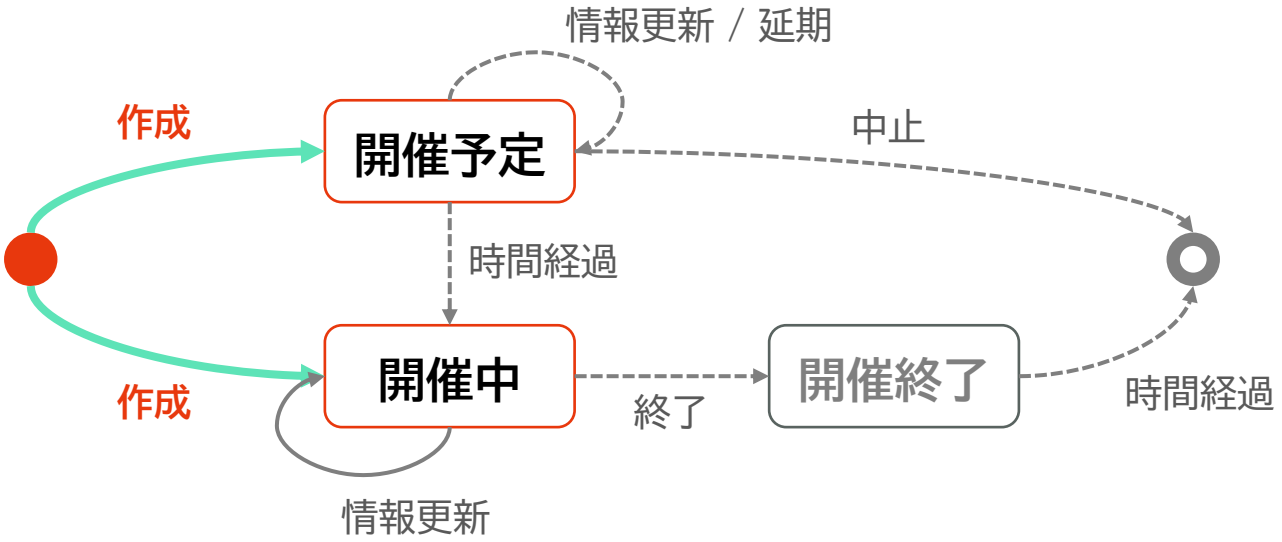
連携システムの本番環境

テスト環境では、データの同期は
*Github Actions*でマニュアル実行。
その際に、どちらの環境に繋げるか指定が可能。
（※本番環境では定期実行ジョブ）

環境の使い分けの方針(1/3)

URBAN HACKS TOKYU CORPORATION

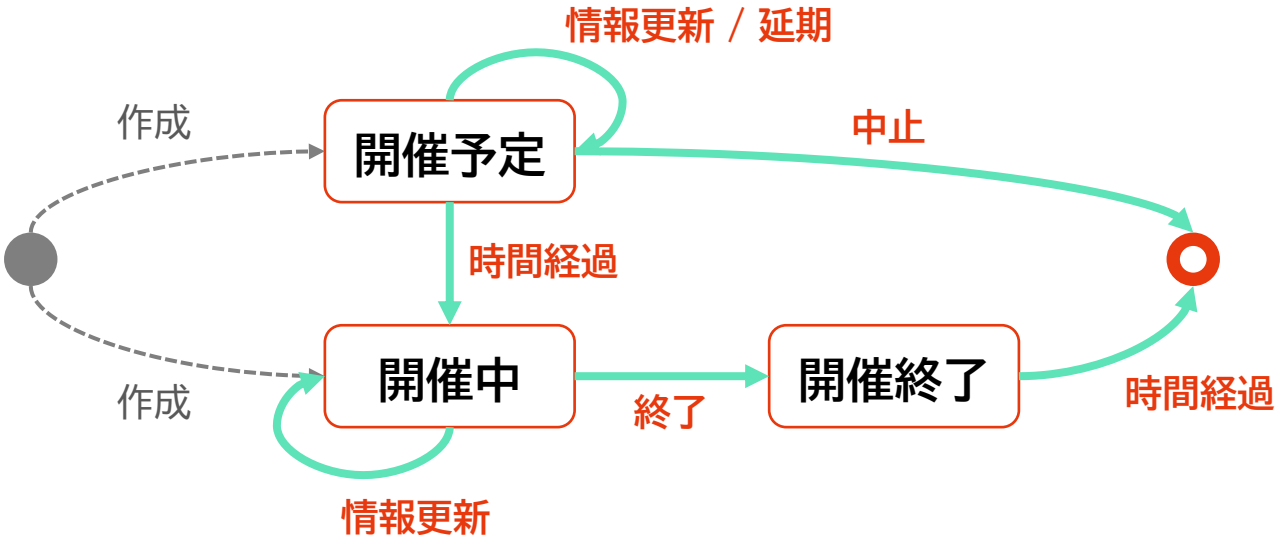
■ モック環境だけでは、連携システム側のデータ構造変化の検知ができなくなるので、初期データ作成パターンは本番環境に接続



環境の使い分けの方針(2/3)

URBAN HACKS TOKYU CORPORATION

■ 以降の状態遷移は、前述の問題解決として、モック環境を利用



環境の使い分けの方針(3/3)

URBAN HACKS TOKYU CORPORATION

■ 結果、テスト目的とカバレッジ両方の網羅を実現

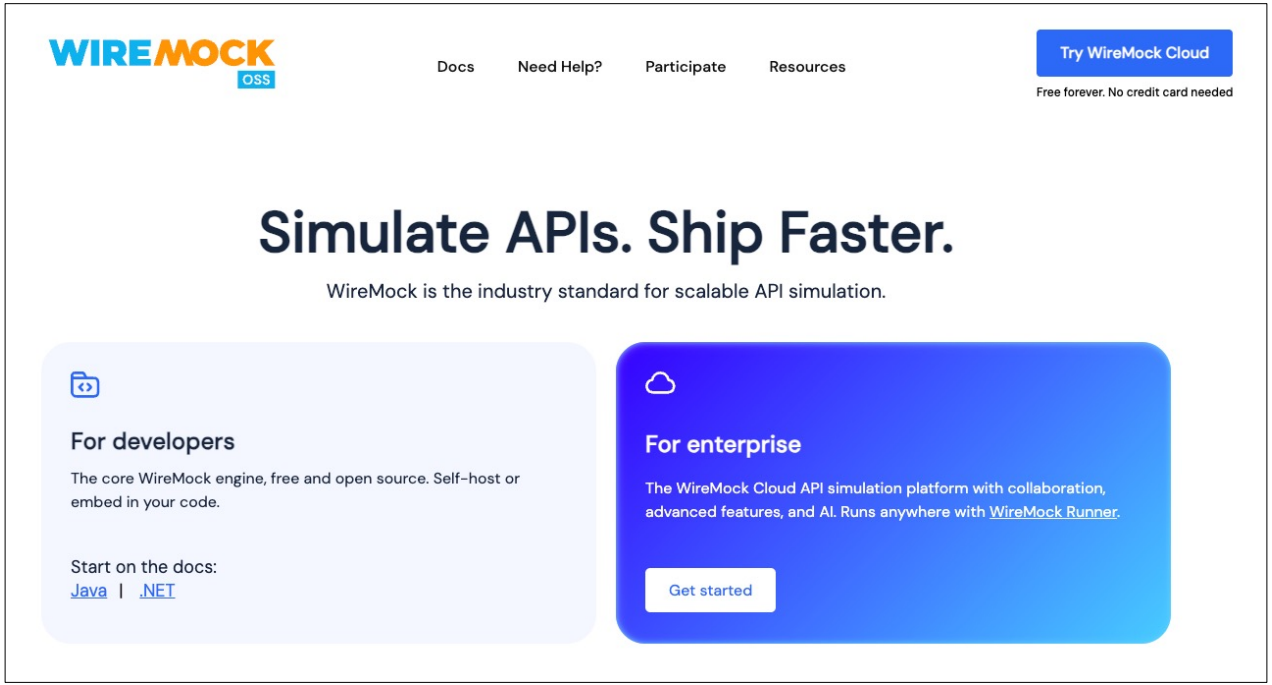
状態遷移	テスト目的	接続先	モック導入前
イベント情報作成	• 外部システムとのデータ構造の整合確認 • 状態遷移の同期機能の確認	本番環境	○
イベント情報更新	• 状態遷移の同期機能の確認	モック環境	△
イベント終了	• 状態遷移の同期機能の確認	モック環境	△
イベント中止	• 状態遷移の同期機能の確認	モック環境	△

モック環境のアーキテクチャ検討(1/5)

URBAN HACKS TOKYU CORPORATION

- 議論の中で以下の選択肢が挙がった
  - Gin(Go言語)等の軽量なWebフレームワークを用いた自作サーバ
  - WIREMOCK等のWebAPIのモック開発に特化したパッケージ利用
  - 組織で包括契約のあるGoogle Workspaceスタックを活用した構成

【参考】WIREMOCK


WIREMOCK
OSS
Docs
Need Help?
Participate
Resources
Try WireMock Cloud
Free forever. No credit card needed
Simulate APIs. Ship Faster.
WireMock is the industry standard for scalable API simulation.
For developers
The core WireMock engine, free and open source. Self-host or embed in your code.
Start on the docs:
Java | .NET
For enterprise
The WireMock Cloud API simulation platform with collaboration, advanced features, and AI. Runs anywhere with WireMock Runner.
Get started

モック環境のアーキテクチャ検討(2/5)

URBAN HACKS TOKYU CORPORATION

■ 前述の課題の観点で比較検討

比較観点	自作サーバ	モック フレームワーク	Google ソリューション
構築工数	△	○	○
維持工数	△	△	○
モックデータのメンテ用意性	△	△	○
セキュリティ	○	○	△

モック環境のアーキテクチャ検討(3/5)

URBAN HACKS TOKYU CORPORATION

- Google App ScriptによるWeb API公開時の制約
  - 本システムが連携する主要システムの認証形式が、HTTPヘッダ内にIDとパスワードを埋め込む方式である
  - Google App ScriptによるWeb APIでは、リクエストのHTTPヘッダの内容を扱うことができない
    - ✓ クエリパラメータの扱いは可能
  - したがって、採用する場合は**認証なしでデータ公開**することとなる

モック環境のアーキテクチャ検討(4/5)

URBAN HACKS TOKYU CORPORATION

- エンジニアメンバーらとも協議をし、以下の観点から、
  初期リリース時点での認証なしでの導入を許容できると意思決定
  - モック環境のURL情報の予測は困難
  - 扱っている情報自体が公開情報であるため、情報流出リスクは極小
  - 連携システム側も十分なセキュリティ(※認証なしのデータ更新は不可能)対策があり、
    データ形式流出リスクも極小

モック環境のアーキテクチャ検討(5/5)

URBAN HACKS TOKYU CORPORATION

■ Googleソリューションによる構築で決定

比較観点	自作サーバ	モック フレームワーク	Google ソリューション
構築工数	△	○	○
維持工数	△	△	○
モックデータのメンテ用意性	△	△	○
セキュリティ	○	○	△→○

AI時代の恩恵

URBAN HACKS TOKYU CORPORATION

■ エンジニアメンバーはGoogle App Scriptを扱うのはほぼ初めて

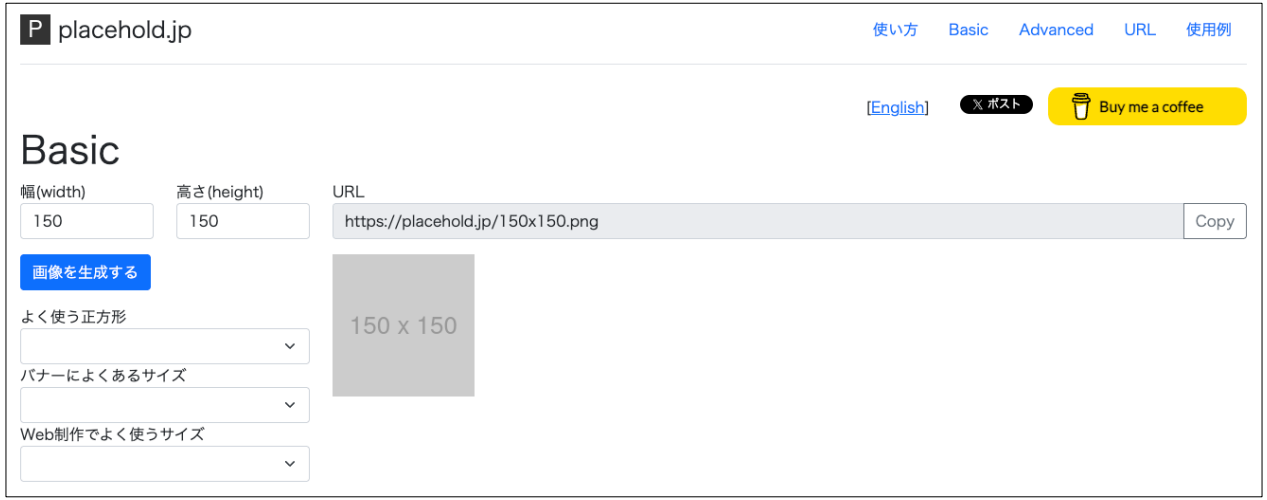
比較観点	自作サーバ	モック フレームワーク	Google ソリューション
構築工数	△	○	○
維持工数	△	△	○
モックデータのメンテ用意性	△	△	○
セキュリティ	○	○	△→○

AI支援によって迅速に開発完了

その他の工夫: 画像の扱い(1/2)

URBAN HACKS TOKYU CORPORATION

- 連携コンテンツとして画像の扱いが必須
  - 画像の準備やホスティングも責務を外部に切り出すアプローチとして、ダミー画像をWeb経由で取得可能なサービス「placehold.jp」を活用



その他の工夫: 画像の扱い(2/2)

URBAN HACKS TOKYU CORPORATION

■ この対応のために、本プロダクトの環境変数の分離を実施

	モック導入前のつくり	モック対応版	
		本番環境へ接続する際	モック環境へ接続する際
環境変数	1つだけ HOST: https://www.example.com	サーバのホストと画像のホストの2つに分離	
		DATA_HOST: https://www.example.com IMAGE_HOST: https://www.example.com	DATA_HOST: https://www.mock.com IMAGE_HOST: https://placehold.jp
データ取得先	https://www.example.com/data.json	https://www.example.com/data.json	https://www.mock.com/data.json
画像取得先	https://www.example.com/image.png	https://www.example.com/image.png	https://placehold.jp/image.png

まとめ

まとめ

【再掲】本日の主題

小規模プロダクトチームが
テスト環境のない外部システムに対して
“十分に使えて維持できる”モック環境を
どう設計したか

モック導入時の課題と対応

URBAN HACKS TOKYU CORPORATION

1. 本番環境・モック環境の使い分けの方針立案
   ➡ *新規作成のテストは本番環境、以降の状態遷移テストはモック環境で*

2. 構築・維持コストの最小化
   ➡ *Google Workspaceを活用したサーバレスの環境構成の実現*

3. QAメンバーのみで対応可能なモックデータの差し替えの仕組み
   ➡ *Google Spreadsheetによる表形式でのデータ管理*

4. 一定のセキュリティ水準の確保
   ➡ *リスク評価のうえ、認証なしで運用開始*

今後の検討事項

URBAN HACKS TOKYU CORPORATION

セキュリティの強化

- 本番環境時の動作に問題ないことを担保したうえで、Google App Scriptで扱えるクエリパラメータにも認証情報を付与し、**モック環境での認証**を実現する

テスト実行の自動化

- 現在は、テスト実行自体は都度データを調整しながらマニュアル実行
- 手順やデータをより定型化し、**日次で自動実行**できる状況を目指す

URBAN
HACKS
TOKYU CORPORATION