クロスプラットフォームフレームワークによるアプリ開発の効率化

Streamlining App Development with Cross-Platform Frameworks

佐々木　諒，増田優生

要　約　アプリケーションを展開するプラットフォームの多様化に対応するためには，複数のプラットフォーム向けのアプリケーション開発が不可欠である．しかし，各プラットフォーム向けに開発すると，開発コストが増え，仕様の統一が困難となり，開発スピードが低下するといった課題がある．このような課題を解決するために，クロスプラットフォームに対応したフレームワークを開発プロジェクトに導入した経緯や成果を述べる．フレームワーク導入後のリリース頻度やプロジェクトのタスク数から，工数の削減や各プラットフォーム間のUI/UXの一貫性確保といった成果が得られた．クロスプラットフォームでの開発効率を向上させるには，要件に対応したクロスプラットフォームフレームワークを採用することが重要である．

Abstract　Developing applications for multiple platforms is essential to respond to the increasing diversity of platforms on which applications are deployed. However, developing for each platform raises issues such as increased development costs, difficulty in standardizing specifications, and reduced development speed. To solve these issues, we have introduced a cross-platform framework into a development project, and this paper describes the background and results of the introduction. Based on the frequency of releases and the number of project tasks after introducing the framework, results such as reduced labor hours and consistent UI/UX across each platform were obtained. To improve cross-platform development efficiency, it is important to adopt a cross-platform framework that meets the requirements.

1. はじめに

アプリケーションを開発・提供するサービスビジネスにおいては，利用者が直面する課題を的確に把握し，それに対する最適な解決策を迅速に提供することが不可欠である．単にアイデアを形にするだけではなく，実際の利用者がどのような価値を求めているのかを見極め，その期待に応えるために，提供する機能を柔軟に調整し続けなければならない．現場から得られるデータやフィードバックを基に設計や仕様を見直し，より良い体験を提供する工夫が事業成長に直結する．

一方で，アプリケーションを展開するプラットフォームは多様化している．iOSやAndroid，Webブラウザといった複数のプラットフォーム向けに同じアプリケーションを提供することが一般的となり，利用者は自分の好みや状況に応じて自由に利用環境を選択するようになった．このような状況下では，どのプラットフォームでも一貫した体験（UX：User Experience）を提供しなければならない．しかし，各プラットフォーム向けに開発を行うと，開発コストや工数が増大し，仕様や機能の統一も難しくなる．結果として，改善の速度や正確性が損なわれ，全体の品質向上が妨げられる．

複数のプラットフォーム向けに開発する方法として，クロスプラットフォームがある．クロスプラットフォームを活用すれば，一つの設計やコードベースから複数の環境向けにアプリケーションを展開できる．そのため，開発リソースの最適化やコストの削減だけでなく，仕様や機能の一貫性を保ちやすくなる．これにより，利用者からの反応を迅速に取り込み，仕様や機能の見直しや改善を効率的に進めることができる．多様な環境で同じ仕様のアプリケーションを動作させることは，現代の事業開発において不可欠な要素であり，競争力を維持するための重要な戦略である．

本稿では，BIPROGY 株式会社（以下，当社）の開発プロジェクトにおける実践経験を基に，クロスプラットフォームの概要や代表的な技術，実際のプロジェクトでの導入経緯，導入による成果と課題について述べる．まず 2 章でクロスプラットフォームの概要および代表的なフレームワークを述べた後，3 章でクロスプラットフォームの導入経緯，4 章で導入による成果と技術的な課題を述べる．

2. クロスプラットフォームの概要

本章ではプラットフォームについて説明して，代表的なフレームワークとその特徴を整理する．

2.1 クロスプラットフォームとは

クロスプラットフォームとは，異なる OS やデバイス環境において，同一の仕様でアプリケーションを動作させる仕組みである．従来，アプリケーションは Windows や MacOS など各プラットフォームそれぞれで個別に開発・保守を行っていた．2010 年代に入り，スマートフォンの普及とともにモバイルアプリケーション開発が急速に拡大したことで，iOS や Android といった異なるモバイル OS への対応が求められるようになった．

これに応じて，さまざまなプラットフォームに対して一元的に開発できるフレームワーク（以下，クロスプラットフォームフレームワーク）が登場し，アプリケーションの開発・保守を効率化する技術が進化してきた．クロスプラットフォームフレームワークとは，複数のプラットフォーム（例：iOS，Android，Web など）向けのアプリケーションを，単一のソースコードで管理・開発する方式を指す．これにより，開発者は個別の OS ごとに異なる言語やツールを習得する必要がなくなり，学習コストや開発期間の短縮が期待できる．また，バグ修正や機能追加といった保守作業も一元的に行えるため，品質の均一化や迅速なアップデートを実現できる．

このように，クロスプラットフォームフレームワークは，複数の環境で一貫した動作を実現し，開発現場における負担を軽減する重要な役割を果たしている．

2.2 代表的なクロスプラットフォームフレームワークの紹介

クロスプラットフォーム開発に利用される主要な四つのクロスプラットフォームフレームワークについて，技術的な特徴を紹介する．

2.2.1 Flutter

Flutter は Google が開発したクロスプラットフォームフレームワークである．Google 独自の Dart 言語を使用し，ウィジェットシステムと呼ばれる独自の仕組みで UI（User

Interface）を構築する．Google の提唱するデザインシステムと親和性が高く，各プラットフォームで一貫した美しいデザインを実現することができる．公式ドキュメント[1]も充実しており，大規模なアプリケーション開発にも対応できる拡張性も持ち合わせている．

2. 2. 2　React Native

React Native は，Meta（旧 Facebook）が開発を主導したクロスプラットフォームフレームワークである．このクロスプラットフォームフレームワークはプログラミング言語として，Web 開発で広く用いられる JavaScript または TypeScript を採用している[2]．そのため，Web 開発者が使い慣れた技術スタックと概念をモバイル開発に直接適用できるため，Web 開発に精通した開発チームにとって学習コストが低いという利点がある．

2. 2. 3　Cordova

Cordova は，HTML，CSS，JavaScript といった Web 標準技術のみでモバイルアプリを構築できる．開発されたアプリケーションは，デバイスに搭載された専用の Web ブラウザ機能（WebView）上で実行されるため，特別なモバイル開発環境を構築せずに済む[3]．WebView 上でアプリケーションを実行するシンプルな構成は，ネイティブ機能へのアクセスにブリッジを介した通信が必要となる．複雑な処理や高度な UI 操作において，実行速度やパフォーマンスの面でネイティブアプリに劣る傾向がある．こういった制約もあり，プロトタイピングやシンプルで処理負荷の少ない小規模なアプリの開発に適している．

2. 2. 4　.NET MAUI

.NET MAUI は Microsoft が提供する最新のクロスプラットフォームフレームワークで，C# や .NET の資産を活用できる．Visual Studio との統合により，開発・デバッグ・デプロイが一貫して行える．Windows や macOS も含めた幅広いプラットフォームに対応し，エンタープライズ向けの堅牢なアプリケーション開発に適している[4]．

2. 3　各クロスプラットフォームフレームワークの比較

各クロスプラットフォームフレームワークは，その技術的特徴や強みに応じて，適したプロダクト開発領域が異なる．また，コミュニティの活発度や開発者シェア率[5]も選定時の重要な指標となる．表 1 に，比較観点ごとの特徴やコミュニティの状況等をまとめる．

表 1　各クロスプラットフォームフレームワークの特徴やコミュニティ状況

フレームワーク	主な特徴	コミュニティ活発度（GitHub スター/ Stack Overflow 質問数）*1	シェア率（2024 年）
Flutter	高パフォーマンス，UI 一貫性，Dart	約 160,000/約 140,000 件	約 46%
React Native	JS/TS 活用，Web 開発者に親和性	約 116,000/約 130,000 件	約 32%
Cordova	Web 技術のみ，手軽，低コスト	約 4,900/約 47,000 件	約 2%
.NET MAUI	C#/.NET 資産活用，企業向け	約 20,000/約 6,000 件	約 11%（Xamarin含む）

各クロスプラットフォームフレームワークはクロスプラットフォーム開発で異なる特徴を持つことがわかる．プロジェクトの要件や開発チームのスキルに応じて選ぶことが肝要である．これらのクロスプラットフォームフレームワーク選定には技術的特徴，コミュニティの活発度，開発者シェア率を総合的に考慮することが重要となる．プロジェクト成功には適切なクロスプラットフォームフレームワークの選定が不可欠である．

3.　当社プロジェクトにおけるクロスプラットフォーム導入の経緯

本章では，クロスプラットフォームフレームワークを採用して，当社が開発および提供しているプロダクト「mierun」[*2]（「みえるん」と読む）の事例を基に，クロスプラットフォームフレームワークの選定理由について述べる．

3. 1　当社プロジェクトでクロスプラットフォームフレームワークを採用した理由について

当社では，保育士の業務時間の見える化や業務負担の軽減を目的として，2022 年 4 月に mierun というサービスを開始した．mierun では保育士向けアプリケーション（以下，保育士アプリ）と保護者向けアプリケーション（以下，保護者アプリ）をリリースしている（図 1）．

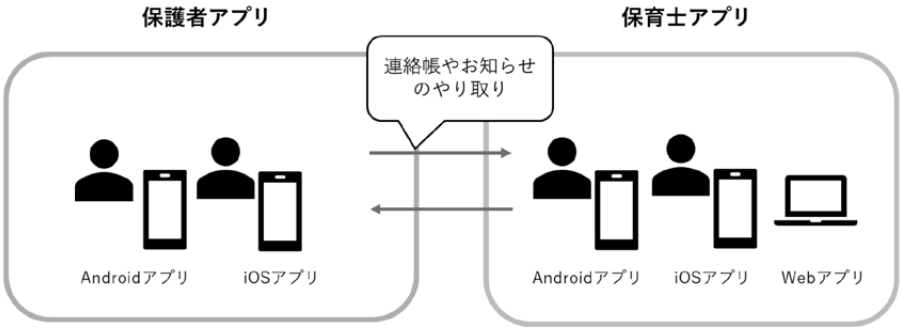

図 1　mierun が提供するアプリケーション

mierun の利用者は保育園に勤めている保育士と園児の保護者であり，所有しているスマートフォンを利用するサービスである．そのため，モバイルアプリケーションを主軸に開発を進めており，iOS と Android の両プラットフォームで同時にアプリケーションを展開しなければならなかった．従来の開発手法では，iOS と Android それぞれの異なるプラットフォーム向けに用意された開発言語で開発するため，同一機能の実装やテストが二重に発生し，工数や進捗管理の負担が大きいことが想定された．さらに，園ごと・保護者ごとに異なるニーズや運用フローに柔軟に対応するためには，仕様変更や機能追加を迅速に反映することが求められる．しかし，プラットフォームごとに個別に開発する体制では，リリース作業や不具合修正のたびに人的リソースが分散し，継続的な開発サイクルの維持が困難になると想定された．さらに，開発チームは新たなメンバーの加入や既存メンバーの離脱によって変動することがあった．経験の浅いメンバーでも迅速に開発へ参加できるよう，学習にかかる時間が少ない開発技術が求められていた．

これらの課題を解決し，開発効率を向上させるためには，単一のコードベースで複数プラッ

トフォームに対応できるフレームワークの導入が不可欠である．次節では，具体的にどのような観点でクロスプラットフォームフレームワークを選定したのかについて述べる．

3.2　Flutter の選定理由

mierun の開発では，クロスプラットフォームフレームワークを選定して，Flutter を採用した．その過程と採用の理由を述べる．

3.2.1　クロスプラットフォームフレームワーク選定の評価軸

本プロジェクトの背景を踏まえ，クロスプラットフォームフレームワークの選定において，以下の三つの主要な評価軸を設定した．

1）デザインの一貫性とパフォーマンス

ユーザーとなる保育園職員と保護者のスマートフォンの利用状況について調査したところ，iOS と Android の利用割合に大きな差はなかった．また，保育園で使われるサービスの仕様上，保護者への園でのアプリの使用説明や問い合わせは開発元ではなく，保育園で受けるということになっていた．そのため，OS 間でデザインや操作性に差異が生じると，対応が複雑化し，保育園側の業務が増加してしまうことが懸念された．そのため，OS 間の差異は極力減らすことが重要な課題となった．以上のことから，クロスプラットフォームフレームワーク選定の評価軸に iOS/Android 間の統一デザイン・操作性，プラットフォーム依存リスクの最小化，アプリの実行速度，といったデザインの一貫性とパフォーマンスを設定した．

2）開発効率と生産性

mierun の開発では園ごと・保護者ごとに異なるニーズや運用フローに柔軟に対応するために，仕様変更や機能追加を迅速に反映する必要があった．そのため，クロスプラットフォームフレームワーク選定の評価軸にテストサイクル短縮，学習コスト・新規参画のしやすさ，といった開発効率と生産性を設定した．

3）技術的持続性とコミュニティ

mierun というサービスの開発を長く続けるために，クロスプラットフォームフレームワークの将来性，開発元のサポート体制，コミュニティの活動度・開発者シェア率などの技術的持続性（サポート体制・コミュニティの活性度や規模）を評価軸に設定した．

3.2.2　クロスプラットフォームフレームワークの比較と選定

前章で紹介した四つの主要なクロスプラットフォームフレームワーク（Flutter，React Native，Cordova，.NET MAUI）のうち，本プロジェクトの要件（高い UI/UX[*3]，開発効率，将来性）に照らして，持続性・コミュニティ規模・実績の観点から，Flutter と React Native の二つに比較対象を絞った．他の二つを除外した理由は以下の通りである．

1）Cordova は WebView ベースでパフォーマンスや UI 一貫性に課題があるが，シェア・コミュニティともに縮小傾向（シェア率は約 2%）である．

2）.NET MAUI は C#/.NET 資産活用やエンタープライズ向けには強みがあるが，コミュニティ規模や実績，クロスプラットフォームの柔軟性で Flutter や React Native には及ばない（シェア約 11%）．表 2 に，Flutter と React Native を評価軸で比較した結果をまとめる．

表 2　概要比較表（オーバービュー）

評価軸	Flutter	React Native
デザイン一貫性とパフォーマンス	◎ iOS/Android 間で一貫性のあるデザイン	○ 一部 UI にデザイン差異リスクがあり
開発効率と生産性	◎ 高速なホットリロード[*4]，マテリアルデザイン[*5] に則った UI 部品を標準で活用できる	◎ 高速なホットリロード，Web 開発経験者にとって学習コストが低い
技術的持続性とコミュニティ	○ コミュニティは成長中の段階 Google のモバイル戦略のコアとして Google I/O 等の公式発表されている	◎ 歴史が長く，コミュニティも成熟している Meta の支援と巨大なコミュニティが強み
開発者シェア率（2021 年）	約 46%	約 32%
コミュニティ活動度（GitHub スター/Stack Overflow 質問数）	約 160,000/約 140,000	約 116,000/約 130,000
サポート企業	Google	Meta

- 評価記号：◎ = 非常に優れる，○ = 優れる，△ = やや劣る

三つの評価軸に対する，Flutter と React Native の評価は以下の通りである．

1）デザインの一貫性とパフォーマンス

Flutter では iOS および Android 間で高いレベルのデザイン一貫性を保証しており，ネイティブアプリに近接した描画性能を提供することができる．プラットフォーム固有のウィジェットを模倣するのではなく，ピクセルレベルで一貫した描画を実現している．

React Native ではネイティブコンポーネントを描画する際に実行時オーバーヘッド（ブリッジ通信コスト）が発生しやすく，特に複雑な UI における高性能の維持に課題が残る．また，コンポーネントがネイティブ OS のデザインに依存するため，プラットフォーム間のデザイン差異リスクを内包する．

2）開発効率と生産性

Flutter では，Google が提唱するマテリアルデザインの部品を標準で提供しているため，UI 部品をゼロから設計・実装する手間が不要となる．これにより，最初からマテリアルデザインで統一された一貫性のあるデザインを短時間で実装することができ，デザイン設計から実装での調整にかかる時間が大幅に削減される．特に，UI がすべて再利用性の高いウィジェットと呼ばれる部品で構成されているため，仕様変更や機能追加が頻繁なプロダクトにおいても，影響範囲を限定して迅速に UI やロジックを更新できるため，高い生産性をもたらす．

React Native では，Web 開発で広く使われている JavaScript/TypeScript を使用するため，Web 開発経験者にとって学習コストが非常に低い点が最大の強みとなる．また多くのライブラリが提供されているため，必要なコンポーネントやソリューションを見つけやすいという利点がある．

3）技術的持続性とコミュニティ（長期視点）

Flutter は，Google が強力に支援している点が大きな強みであり，モバイルに留まらず，Web，デスクトップ，組み込みへの展開も積極的に進行中である．コミュニティの規模は React Native と比較すると現在も成長段階にあるものの，Google のモバイル戦略のコアとして位置づけられており，Google I/O などの公式発表においても今後の投資継続が明言されている．このため，クロスプラットフォームフレームワークが将来的に撤退するリスクは低いと判断できる．

React Native は，開発元である Meta の強力な支援に加え，巨大で成熟したコミュニティを持っていることが最大の強みである．歴史が長く，コミュニティがすでに成熟しているため，問題解決や情報共有が他言語のコミュニティに比べて容易であるという優位性を持っている．

3. 2. 3　選定結果（Flutter の採用）

ここまでの比較分析の結果，「開発・テストサイクルの短縮」「高性能で一貫したユーザー体験」という主要目標への貢献度が高いと判断し，Flutter を採用した．

この選定結果に基づいて，本プロジェクトのモバイルアプリケーション開発は Flutter および Dart 言語を採用した．

3. 3　Flutter on the Web の導入について

前節の選定理由から mierun では開発で使用するクロスプラットフォームフレームワークを Flutter，バックエンドには Firebase を採用し，2021 年 4 月から MVP 開発[*6] を開始した．また，開発中に，園長先生といった園の管理者が mierun を利用する際に PC などの大きい画面で情報を一括で確認したいという意見があった．そのため，2021 年 11 月から保育士アプリ・保護者アプリとは異なる新たなアプリケーション（以下，園管理システム）の開発に着手した．園管理システムでは，出欠状況の確認やお知らせ送信機能といった基本的な機能のみを対象として，リッチな UX は不要と判断した．特定の要件がなかったため，社内標準である Vue.js と Spring Boot の構成を採用し，インフラに Azure を採用した．そして 2022 年 4 月から保育園での本格的な運用を開始した．この時点での mierun の技術構成図は以下の図 2 である．

運用を進める中で，保育士から，園で利用する端末が PC のみであり，mierun を利用することができないといった意見があった．そのため，PC のブラウザ上でも保育士アプリと同等の機能を利用できる環境が求められた．また，園管理システムについても追加機能の開発が求められる状況となった．しかし，従来の Vue.js および Spring Boot による技術構成では，モバイルアプリケーションと同一機能一式を Web 上で実現する場合，開発コストおよび提供までの時間が過大となることが想定された．これを踏まえ，園管理システムを保育士アプリへ統合し，モバイルおよび Web 双方の開発基盤を統一する方針を検討した．その手段として Flutter on the Web（以下，Flutter Web）の導入が有力な選択肢として挙げられた．

Flutter Web とは，モバイルアプリケーションと同じコードベースを活用し，Flutter で Web アプリケーションを構築するための技術である．Flutter プロジェクト内で Flutter Web に対応するための設定を行うだけで，モバイルアプリケーションの既存のコードから Web アプリケーションを作成することができる．Flutter Web も Dart で開発できるため，新たな言

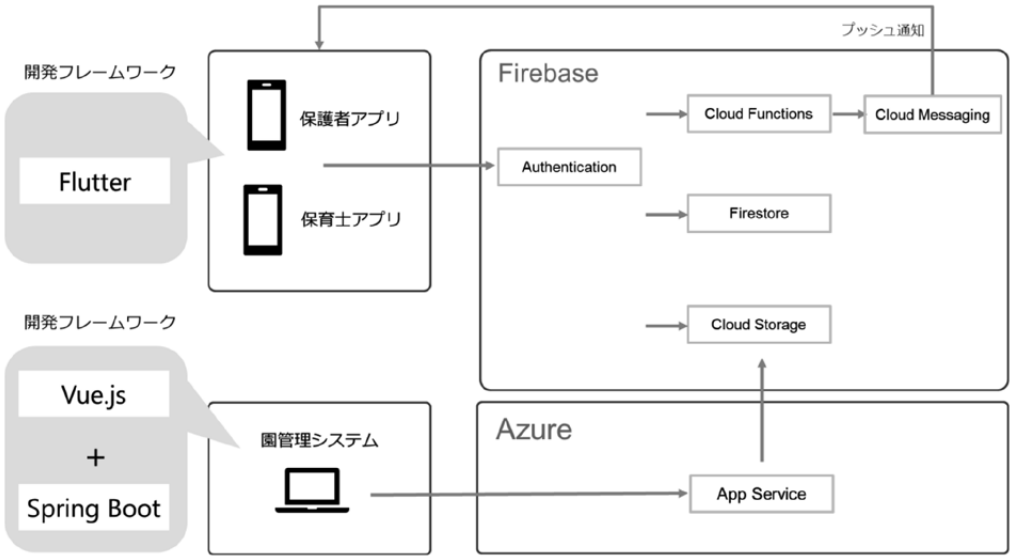

図 2　開発当初の技術構成図

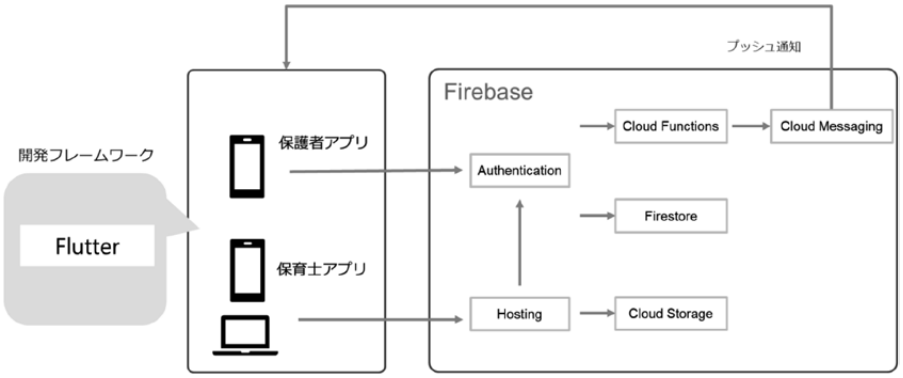

図 3　Flutter Web 導入後の技術構成図

語の学習は不要であり，モバイルと同様の機能・デザインを Web でも実現できる．mierun においてもモバイルアプリケーションの既存実装を再利用することで，より効率的な開発ができると想定した．以上のことから，mierun の開発に Flutter Web を導入し，開発基盤を Flutter で統一した．Flutter Web 導入後の mierun の技術構成を図 3 に示す．

4.　導入の成果・課題

本章では，mierun の開発基盤を Flutter に統一したことで得られた成果とクロスプラットフォームフレームワーク特有の技術的課題について述べる．

4. 1　成果

本節では，コード量・バックログの削減，開発効率の向上，UI/UX の一貫性という観点から得られた成果を述べる．

4.1.1　コード量・バックログの削減

Flutter で開発基盤を統一したことにより，mierun 全体のソースコード量を削減できた．前章で述べた通り，開発基盤を Flutter に統一する前は Vue.js，Spring Boot も採用していた．そのため，同じ機能であっても複数の技術スタックで重複して実装・保守することになり，全体のコード量が増加していた．表 3 は Flutter Web を導入する前の保育士アプリ・保護者アプリ，園管理システムの開発で実装したそれぞれのソースコードの行数である．この時期では保育士アプリはおよそ 100,000 行，保護者アプリはおよそ 67,000 行，園管理システムはおよそ 5,000 行のソースコードが存在していた．これらを合計すると，mierun 全体の総コード量はおよそ 172,000 行に達していた．

表 3　Flutter 統一前のソースコードの行数

システム名	ソースコード行数（行）
保育士アプリ	約 100,000
保護者アプリ	約 67,000
園管理システム	約 5,000

Flutter Web の導入によって，園管理システムの主要な機能を保育士アプリ側に統合し，Web アプリケーションも Flutter で提供するようになった．その結果，Vue.js と Spring Boot で構築されていた園管理システムのフロントエンド・バックエンド部分が不要となり，これらのソースコードを削除することができた．一方で，保育士アプリ側では Web 対応に伴い，コードが追加されたが，Flutter 統一後の mierun 全体の総コード量はおよそ 168,000 行となった．園管理システムのソースコードが削減されたことで，同じ機能の開発や運用にかかる負担が減った．

また，Flutter に統一したことでプロジェクト内のタスクも整理された．mierun では開発の作業をプロダクトバックログアイテム[*7]（以下，バックログ）として管理している．Flutter 統一前は，保育士アプリ・保護者アプリ・園管理システムそれぞれにバックログが存在し，2022 年 12 月時点で全体のバックログは 432 件あった．その中で園管理システム向けの未着手バックログは 64 件存在した．当時の開発メンバーは 3 名であり，1 カ月で 14 件のバックログを完遂できていたため，64 件のバックログを完遂するには，机上の計算では約 13.7 人月の工数を要する．Flutter Web 導入後は園管理システムを独自に開発することがないため，これらの園管理システム向けのバックログが不要となった．その結果，管理すべきタスクを整理でき，園管理システムの開発で想定していた工数も削減できた．

このように，Flutter による開発基盤の統一は mierun 全体のソースコード量とバックログの両面での削減を実現した．複数の技術をまたいで同じ機能を実装・保守する必要がなくなり，開発や運用にかかる負担が軽減されたことで，開発チームは機能改善や新たな機能の開発により多くの時間を割けるようになった．

4.1.2　開発効率の向上

Flutter では一つのソースコードから複数のプラットフォーム向けにアプリケーションを同

時に展開できるため，機能開発やテストが一度の実装で済むようになった．例えば，保育士や保護者からのフィードバックや新たな要望があった場合も，ソースコードを一箇所のみ追加・修正するだけで，Android・iOS・Web のすべてのプラットフォームに即座に反映できた．この仕組みによって，mierun の継続的なアップデートを実現した．

mierun では 2022 年 4 月のサービス開始以降，新機能を継続的にリリースしており，2022 年 12 月からは Flutter Web による Web アプリケーションの提供も開始した．以下の図 4 は 2022 年 12 月から 2023 年 5 月までの mierun のリリース日と次回のリリースまでに経過した日数を示したものである．この期間内にリリースは 14 回行われており，mierun では長期休暇や年度末を除いて，リリース頻度を約 2 週間に 1 回というペースを維持している．エンジニアの転職や組織の生産性向上を支援している Findy 社の生産性調査のレポート[6]では，リリース頻度が月に 1 回以上の企業の割合は全体の 34.2%であり，mierun のリリース頻度は高い水準にある．短期間でのリリースを繰り返すことで，ユーザーから寄せられた要望や不具合の修正，新機能の追加を迅速に反映でき，mierun の品質向上とユーザー満足度の向上を同時に実現した．

図 4　リリース日と次回のリリースまでの経過日数

4. 1. 3　UI/UX の一貫性

開発基盤を Flutter で統一する前は，同じ機能や画面を異なる技術で実装しており，UI や UX の統一には多くの課題があった．例えば，Vue.js で作成した画面と Flutter で作成した画面では，ボタンや入力フォーム，リスト表示などの細かなデザインや動作が微妙に異なり，園管理システムの画面のデザインやコンポーネントの仕様を保育士アプリ側に揃えるために，実装の段階で調整や確認作業が発生し，開発に時間がかかっていた．実際，Vue.js と Flutter の両方で実装する場合，修正や機能追加のたびに両方のコードをメンテナンスすることになる．こうした状況は，開発チームにとって大きな負担となっていた．

mierun では，アプリケーションのデザインに 3. 2. 2 項で紹介したマテリアルデザインを採用した．Flutter にはマテリアルデザインに準拠した UI 部品を標準で提供しているため，機能を追加したり画面を改善したりする際にも，これらの UI 部品を組み合わせることで，異なるプラットフォーム間でも統一感のある外観や操作性を容易に実現できた．さらに，Flutter Web の導入によって，レスポンシブデザインの実装も容易になり，すべてのプラットフォー

ムでUIや見た目を統一できるようになった．その結果，ユーザーがスマートフォン，タブレット，パソコンなど異なるデバイスを利用しても，mierunを利用できる環境を提供できた．図5はモバイルアプリケーション向けの画面とWebアプリケーション向けの画面例である．これら二つの画面を比較しても，ユーザーの利用する端末に合わせて最適な表示を行うことで，どのデバイスからでも見やすく，使いやすい画面を実現した．

図5　各プラットフォームの画面例

4.2　クロスプラットフォームフレームワーク導入後の課題

本節では，mierunの開発基盤をFlutterに統一した結果として生じた技術的な課題について述べる．

4.2.1　プラットフォーム固有の挙動対応と抽象化の限界

これまで述べた通り，クロスプラットフォームフレームワークは複数のプラットフォーム向けに単一コードでアプリケーションを生成できる利点を持つ．しかし，各OSやバージョン，デバイスに起因するネイティブな挙動の差異を完全に吸収することは困難であり，特定機能の実装においてプラットフォーム固有の対応が求められた．これは，クロスプラットフォームフレームワークが提供する共通の抽象化レイヤーの範囲外にある機能に対応する必要があるためである．具体的には，OS固有機能の利用やデバイス固有のハードウエア（カメラや端末内ファイルなど）への直接的なアクセス，さらにはプラットフォームごとに異なるセキュリティおよびプライバシーポリシーへの準拠が求められる．参考までにプロジェクト内での対応について述べる．

1）権限管理と各OSベンダーへのセキュリティポリシー対応

モバイルのデバイス機能へのアクセスに関するセキュリティポリシーはOS固有のセキュリティモデルに依存するため，iOSとAndroidそれぞれでネイティブ対応が不可欠となった．具体的には権限リクエストの目的をアプリケーションのメタデータファイル（iOSのInfo.plist，AndroidのAndroidManifest.xml）に記述することや，機能によってはアプリ内で各機能の使用前にユーザーへ利用許可を求める挙動等が義務付けられている．これらの不備はアプリケーションのクラッシュやストア審査でのリジェクト要因となる．これに加え，OS

ベンダーはプライバシー保護のため，セキュリティポリシーを変更することがある．この変更は設定ファイルの変更だけでなく，アプリ機能の改修につながるような大きな対応が求められることもあった．

そのため mierun においては，Apple や Google の公式情報を継続的にキャッチアップすることをチームで実施している．また，実際にポリシーが変更された際は，開発スケジュールの中に事前検証と対応をプロジェクトタスクとして明示的に組み込むことで対策した．

2）Web ブラウザにおけるモバイルとは異なる制約への対応

Web ブラウザでは，ユーザーの利便性とデータ通信量への配慮からモバイルとは異なる規格の制約がいくつかある．その中の一つに，音声付きの動画等の自動再生の制限[*8]がある．iOS/Android では権限が付与されていれば音声・動画のプレビュー表示が自動で開始されるのが一般的である一方，Web ブラウザではユーザー操作（クリックなど）を伴わない再生は，この制限によってブラウザ側でブロックされ再生が開始できない事象が発生した．

mierun ではプレビュー画面の描画を開始する前に，ユーザーに「再生」や「プレビュー開始」を促す UI ロジックを追加することで対応した．mierun の UI 設計は元々モバイルアプリベースで行っていたが，3 章で述べた Flutter Web の導入後は，このように Web ブラウザとモバイルで両立できるユーザー体験を設計することでプラットフォーム間の差異を減らした．

以上のように，クロスプラットフォーム開発では各プラットフォームのセキュリティポリシー改定への継続的な監視や，Web 環境特有の技術的制約を考慮した統一的な UI/UX 設計の採用といった，プラットフォーム固有の課題に対応するプロセスを組み込むことが求められる．

4. 2. 2　サードパーティライブラリとネイティブ環境の制約

Flutter を用いたアプリケーション開発では，サードパーティライブラリ（追加機能パッケージ）の活用によって，開発効率および機能拡張性の向上が期待できる．これらのライブラリは，カメラや位置情報，通知機能など，端末のハードウエアや OS が提供する各種機能にアクセスする役割を担う場合が多い．

サードパーティライブラリがこうした端末機能やアプリケーションへアクセスする仕組みには，主に二つの実装パターンが存在する．一つは Flutter などクロスプラットフォームフレームワークが提供する抽象 API を利用する方法であり，もう一つはサードパーティライブラリ自身が直接的に iOS や Android のネイティブ API を呼び出す方法である．後者の場合，サードパーティライブラリの更新や OS 固有の仕様変更への対応が，Flutter 本体やアプリケーション本体よりも遅れるというリスクがある．

Apple や Google などのプラットフォーム提供元は，ユーザーのプライバシー保護やセキュリティ強化のため，新しい要件や厳しい規制を継続的に導入している．そのため，サードパーティライブラリの対応が遅延した場合，アプリケーション本体が新要件に対応済みであっても，依存するライブラリが未対応であることを理由に，アプリがストア審査を通過できない．このように，サードパーティライブラリの対応状況がアプリケーションのリリース計画におけるボトルネックとなるリスクが常に存在する．サードパーティライブラリの対応遅延がもたらすリリース延期のリスクを図 6 に示す．

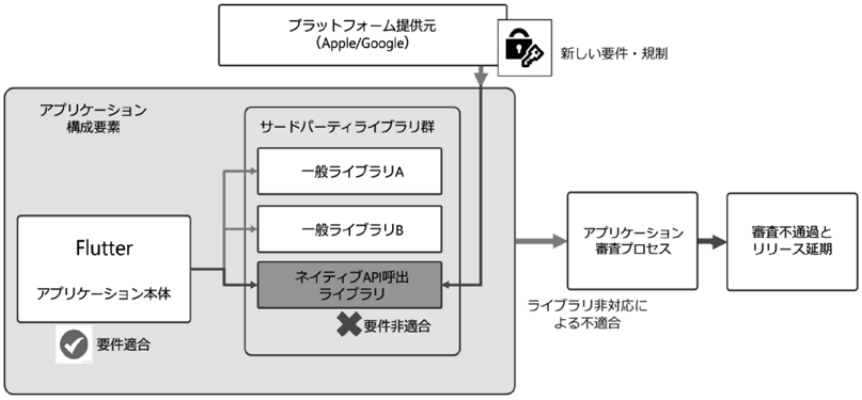

図 6　サードパーティライブラリの対応遅延がもたらすリリース延期のリスク

具体例として，Apple が導入したプライバシーマニフェスト（Apple Privacy Manifests）[*9] が挙げられる．これは，アプリケーションが利用するすべてのライブラリが Apple の定めるプライバシー基準に適合していなければ，App Store での公開が認められないという要件である．mierun プロジェクトにおいても，一部のサードパーティライブラリが新基準への対応を完了していなかったため，アプリケーション本体の修正を終えた後も，ライブラリ側のアップデートを待つ事態が生じた．リリーススケジュールへの影響を抑制するため，該当ライブラリを代替可能なものへ切り替える措置を講じた．

この種のリスクを低減するため，サードパーティライブラリの選定に際しては，以下の基準を重視している．第一に，ライブラリの開発および保守が継続的に実施されていることを確認する．具体的には，GitHub 等のリポジトリにおけるコミット履歴や Issue 対応状況を調査し，主要な OS アップデートや新規制への追従状況を評価する．第二に，位置情報，カメラ，通知等，プライバシーやセキュリティに関わる機能を持つライブラリについては，単なる機能性のみならず，各プラットフォームの標準 API への依存度や，将来的な規制変更への対応力も事前に検証する．複数の代替ライブラリを適宜比較して，最適なものを選定する．

これらの選定基準を適用することで，外部要因による開発停滞や品質低下のリスクを抑制し，安定したサービス提供を維持している．

5. お わ り に

本稿では，サービスビジネスにおける多様なユーザー環境への対応と開発効率を向上させる手段として，クロスプラットフォームフレームワークによるアプリケーション開発の効率化について，mierun プロジェクトの事例を基に考察した．

mierun では，一貫した体験の提供や仕様変更を短いサイクルで実現できる体制が求められていた．これらの要件に対し，「デザインの一貫性とパフォーマンス」「開発効率と生産性」「技術的持続性とコミュニティ」という三つの評価軸を設定し，代表的なクロスプラットフォームフレームワークを比較・検討した．その結果，「開発・テストサイクルの短縮」「高性能で一貫したユーザー体験」といった観点が mierun の目標達成に貢献すると判断したため，Flutter を採用した．加えて，園管理システムの機能の拡張に伴い，開発と運用の負担が大きくなっていくことが想定されたため，Flutter Web も採用した．その結果，開発効率の向上，コード量

やバックログの削減，UI/UX の一貫性確保といった成果が得られ，開発チームの負担が削減された．一方で，プラットフォーム固有の挙動への対応や，サードパーティライブラリとネイティブ環境に起因する課題も明らかとなった．これらの課題に対する対策として，Web ブラウザとモバイルで両立できるユーザー体験の設計やライブラリの選定基準を設けた．

サービスビジネスにおいては，多様なユーザー環境や変化するニーズに柔軟に対応しながら，プロダクトの継続的な改善が求められる．競合製品との競争に打ち勝ち，ユーザーに選ばれ続けるためには，魅力的な機能を迅速に実装し，プロダクトの価値を絶えず高めていかなければならない．プロダクトの価値を高めるには開発効率の継続的な向上が不可欠であり，チーム全体で効率的な開発体制を構築することで，チームはプロダクトの機能の開発や改善に集中できる．クロスプラットフォームフレームワークはプロダクトの開発効率を向上させる選択肢の一つであり，プロジェクトの要件に適したフレームワークを採用することが重要である．

本稿がクロスプラットフォームフレームワークについて知るための一助になれば幸いである．

最後に，本稿の執筆にあたり，多くの助言とご指導をいただいた関係者の皆様に深く御礼申し上げる．

＊ 1　記載している GitHub スター・Stack Overflow 質問数は 2024 年 10 月時点のものを統計した数値である．
＊ 2　mierun の詳しい説明：https://www.biprogy.com/solution/service/mierun.html
＊ 3　ユーザーが直感的な操作（UI）でストレスなく利用でき，快適さや満足感（UX）を得られる状態．
＊ 4　コードの変更を瞬時にアプリケーションの実行画面に反映できる機能．再コンパイルやアプリケーションの再起動が不要なため，UI の調整やバグ修正の対応時間が劇的に短縮され，試行錯誤の時間が大幅に削減される．
＊ 5　Google が提唱するデザインシステム．物理的な法則（影や奥行きなど）をデジタルインターフェースに取り入れ，視覚的な一貫性と快適な操作感を提供する．Flutter はこのガイドラインに準拠したウィジェット（UI 部品）を標準で搭載しており，デザイン工数を大幅に削減した効率的な開発を可能にしている．
＊ 6　最小限の機能だけを備えた製品を開発し，フィードバックを得ながら改良を重ねていく手法である．
＊ 7　チームがプロジェクトに取り組むタスクである．
＊ 8　Chrome の自動再生ポリシー：https://developer.chrome.com/blog/autoplay?hl=ja
＊ 9　ユーザーを追跡して集めたデータの透明性を高め，プライバシー保護を強化することを目的に Apple が義務化した．https://developer.apple.com/documentation/bundleresources/privacy-manifest-files

参考文献

［1］ Flutter，Google LLC，https://flutter.dev/
［2］ React Native，Meta Platforms, Inc.，https://reactnative.dev/
［3］ Apache Cordova，Apache Cordova，https://cordova.apache.org/
［4］ .NET MAUI とは，Microsoft Corporation，https://learn.microsoft.com/ja-jp/dotnet/maui/what-is-maui?view=net-maui-9.0
［5］ Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2023，statista，2025.07，https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/
［6］ ファインディ株式会社，ソフトウェア開発における「開発生産性」に関する実態調査レポート，2025 年 7 月 1 日，https://findy.co.jp/3036/

※　上記注釈および参考文献に記載の URL のリンク先は，2026 年 1 月 23 日時点での存在を確認．

執筆者紹介　佐々木　　諒（Ryo Sasaki）

2021年日本ユニシス(株)入社．アジャイル推進室に所属し，アプリケーション開発を担当．mierunの開発・運用に携わった後，AIによるコード生成・レビューツールの開発を担当．

増　田　優　生（Yusei Masuda）

2019年日本ユニシス(株)入社．アジャイル推進室に所属し，アプリケーション開発やアジャイル開発を推進．現在はスクラムマスターとしてmierunの開発・運用を担当．