Red Hat

API オーナーのマニュアル

成功を収める API チームの 7 つのベストプラクティス

Manfred Bortenschlager (マンフレッド・ボーテンシュラガー)

Red Hat API ベース統合ソリューションおよび API 管理ビジネス開発担当取締役

Andrew Mackenzie (アンドリュー・マッケンジー)

Red Hat ソフトウェアエンジニアリング API 管理担当取締役

Jaya Baskaran (ジャヤ・バスカラン)

Red Hat プリンシパルテクニカルマーケティングマネージャー

Greg Pack (グレック・パック)

Red Hat アプリケーションサービスおよび API 管理ソリューションシニアプロダクトマーケティングマネージャー

目次

エグゼクティブサマリー

5G やエッジコンピューティングなど、先進的なアプリケーション開発、接続性、インフラストラクチャにおける発展により、ビジネスプロセスが生成される方法や場面は変わり続けています。こうした発展によってビジネスの速度は増すばかりですが、先進的なアプリケーション・プログラミング・インタフェース (API) がなければ、そのメリットを手に入れることはできません。

現在、API は先進的な組織をデジタルでつなげる手段となり、運用や製品からパートナー戦略まで、あらゆるものに新しい機能を追加しています。今すでに API 戦略がある場合でも施行を準備している場合でも、この e ブックは、API プログラムを成功させるための 7 つのベストプラクティスをチームに提供します。

はじめに

この e ブックをお読みになる前に、自社の API プログラムの主な目的をよく考えてみてください。本書の中で後ほど示す質問の中から、目的の変更、検証、内容の充実に役立つヒントが得られることでしょう。

効果的な API プログラムとは、組織全体をカバーする企業戦略をベースに構築し、企業の目的達成を補助するものです。次の 4 つの質問に答えを出せたそのとき、優れた戦略を確立できるようになります。

1. なぜ API を実装するのか (理由)
2. API の使用が見込まれるのは誰か (対象)
3. 実装した API から具体的にどのような成果を得たいのか (目的)
4. 成果を達成するために、API プログラムをどのように実行する予定か (方法)

なぜ API を実装するのか (理由)

API を検討する際には、その価値と目的を総合的に理解することが大切です。潜在的なビジネス価値を最大化するためには、よくある誤解を回避することが重要です。一般的な誤解とは、API はユーザーから使用料を回収できる場合にのみ価値があるというもので、これは API が製品である場合には間違いではありません。しかし、API プロジェクトの多くは内部向けであり、市場投入時間、再利用性、売上、ブランド認知度、アフィリエイトの参照回数など、多様なメトリクスを生成します。

代表的な API プロバイダーのユースケースとして、以下の 5 つが挙げられます。

1. **オムニチャネルのスケーリング：**モバイルアプリや IoT (モノのインターネット) などの多様なチャネルを通じてアクセシビリティを拡大します。
2. **エコシステムの拡大：**顧客またはパートナーのエコシステムを育成し、拡大を助けます。
3. **リーチの拡張：**広範なトランザクションまたはコンテンツ配信ネットワークを確立します。
4. **新しいビジネスモデルの強化：**まったく新しいビジネスモデルによってイノベーションを促進します。
5. **内部のイノベーションの創造：**組織内でイノベーションを創造します。

成功する API 戦略は、Amazon、Salesforce、Twilio などの先進的なテクノロジー企業で多数確認できます。だからと言って、これらの主要テクノロジー企業だけが API を使用して収益化、イノベーション、パートナーシップの成長、アジリティを獲得できるということではありません。多くの歴史ある企業には従来からのシステムが多数あり、これらがすぐに消えることはないでしょう (おそらくは絶対になくなりません)。API はこうした従来のシステムを統合するための優れた接着剤となり、これらを内部または外部に公開し、他のサービスとマッシュアップして新しい価値を生み出します。デジタル・トランスフォーメーションを進めるどの企業も、適切に構成された API 戦略を実装することで、 API エコノミーのメリットを活用できます。

API へ投資するという決定を組織に納得させるには、この「理由」の部分が十分に強固でなければならないことは明らかです。

API の使用が見込まれるのは誰か (対象)

API のエンドユーザーが誰であるのかを理解すると、API プログラムの成功メトリクスの定義に役立ちます。

- **内部エンドユーザー：**内部チームが API を使用して、複数のビジネス目的のために企業の情報にアクセスできます。
- **外部エンドユーザー：**同じ API またはそのサブセットを、外部のサードパーティ開発者が自社アプリケーションで使用することができます。

API へ投資するという決定を組織に納得させるには、この「理由」の部分が十分に強固でなければならないことは明らかです。

実装した API から具体的にどのような成果を得たいのか (目的)

API によってもたらされる具体的な成果を確認するには、組織内外の視点を考えてみます。

- **内部からの視点：**独自で価値のあるアセットを使用します。Meta の「いいね！」データなど、他にはないデータを保有する組織は、価値の高い API を提供できます。
- **外部からの視点：**市場のダイナミクス、動向、競合他社、顧客の行動を考慮します。外的な力はビジネス戦略を形成する要因となり、API の機能に影響を与えます。

市場のダイナミクスは地図 API に多大な影響を与えます。初期の地図作成企業は地図をリアルタイムに表示したいというニーズを見過ごし、それが Waze などのスタートアップ企業の隆盛につながりました。Google は Waze を買収し、そのテクノロジーを API に統合して成功させました。Twitter、Reddit、およびその他の巨大企業も API を取り入れて、イノベーションやエコシステムの成長を実現させました。

適切な API 戦略では、内部アセットと市場の外部インサイトを組み合わせることがよくあります。

成果を達成するために、API プログラムをどのように実行する予定か (方法)

最後の質問、「成果を達成するために、API プログラムをどのように設計するのか」とは、実装と実行に関するもので、以下のことを入念に計画する必要があります。

1. **技術面での選択：**API を構築するためのテクノロジースタックを決定します。
2. **設計とメンテナンス：**API の設計およびメンテナンスの戦略を計画します。
3. **プロモーションとマーケティング：**組織内で API を普及させ、外部に公開します。
4. **リソース割り当て：**API の開発とメンテナンスに必要なリソースを割り当てます。
5. **チームの編成：**API の開発と管理の能力を持つチームを結成します。
6. **開発者コミュニティ：**内外の開発者向けの専用コミュニケーションプランを作成し、維持します。
7. **成功メトリクス：**ビジネス目標に対する成功度を追跡する手法を確立します

API の理由、対象、目的、方法を明らかにすると、組織は進化する API エコノミーにおいてイノベーションと成長を促進できます。これらの質問への回答は組織によって異なり、目標、導入する戦略、そして最も重要な要素として API チームの影響を受けます。

API チーム

API チームは通常、その他のプロダクトチームと同様の構成になっています。顧客が社内であっても社外であっても、API チームは他のユーザーが利用するインフラストラクチャを構築、デプロイ、運用、最適化する責任を負っています。

プロダクトチームと同様に、API チームもさまざまな人材から構成されています。チームには、戦略と目標の達成に向けて舵を取る製品中心の担当者、API 設計のベストプラクティスを保証する設計重視のチームメンバー、API テクノロジーをコーディングするエンジニア、最終的に API を実行する運用担当者が参加します。作業を進めていくうちに、サポートおよびコミュニティのチームメンバー、API スペシャリスト、セキュリティ担当者など、他のメンバーを追加することもあります。拡張した API チームには、開発者コミュニティのメンバーも含めることができます。

このチームは大所帯になることもありますが、特に小規模組織では、一部の個人が複数の役割を担当することがあります。すべての関係者の意見を反映させることが重要です。チームメンバーが懸念事項を確認する程度のものであっても構いません。

多くの場合、API チームは一時的に結成され、異なるラインマネージャーが担当するさまざまな組織単位に属しています。このような構造では、API に対する共有ビジョンの定義が極めて難しくなります。大規模な API プログラムでは、さまざまな API チームがコラボレーションする必要があります。

組織の規模にかかわらず、この e ブックに記載した 7 つのベストプラクティスは、成功する API チームの確立に役立ちます。API チームには、最終的には予想していたよりも多くの人が参加することになる可能性があります。

ベストプラクティス #1：API の価値にこだわって集中する

API プログラムは、価値をもたらすと同時に複雑さを回避するという中心的な目標を優先する必要があります。価値提案は、ユーザーの使い勝手を左右します。ユーザーの使い勝手は、API を成功に導く重要な要因です。効果的なマーケティングには、説得力のある価値提案が欠かせません。企業の目的と一致させることで持続可能性がもたらされ、既存の企業は API を通じてオファリングを強化できるようになります。

Alex Osterwalder の API の価値モデルは、ユーザーのメリットを API の機能に一致させ、ユーザーのニーズと API の価値との重要な適合を作り出しています。[1] ニーズに対処し、課題を軽減し、価値を創成することが大切です。

API にとっての意味

このプロセスは反復的なもので、その最初のステップでは、ユーザーが完了しようとしているジョブを記述します。これには、非常時に緊急連絡の自動送信、重要なファイルのバックアップ、特定のイベントを検出するためのデータのサンプリングなどがあります。

2 番目のステップでは、ジョブを実行しようとする前、実行中、実行後にユーザーに影響を与える、特定の弱点 (ペイン) を突き止めます。これには、信頼できる複数回試行のメッセージ送信、障害の検出、複数メッセージの管理、位置ベースのメッセージシステム統合、最小の帯域幅使用によるファイル提供の保護、大量のデータのリアルタイムの照合などがあります。

ユーザープロファイル構築における 3 番目のステップでは、その他の種類の通知 (脅威について警告するのではなく、信頼性が十分な場合はその他のストレージ機器を排除する、イベントに基づいてアクションを自動的にトリガーするなどによりチャンスを醸成する) など、潜在的なメリットの概略を説明します。

価値マップに目を移すと、API の機能、特徴、サービスは、ペインリリーバーとゲインクリエイターに注目してマッピングする必要があります。このプロセスから、メッセージ配信を行うメッセージング API、効率的なバージョン更新のためのストレージ同期 API、構成可能なデータストリームを提供するデータ集約 API など、具体的な事例が形成されます。

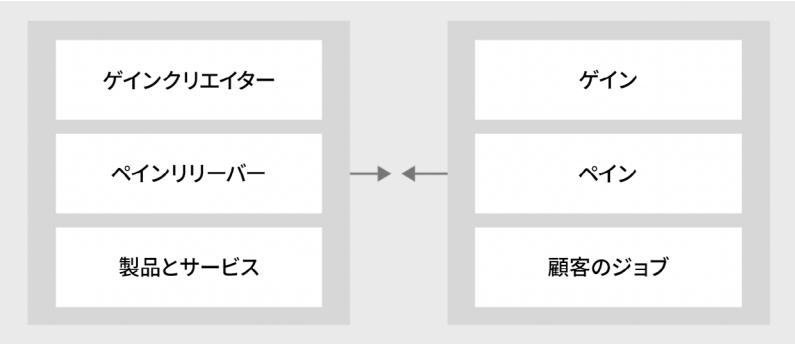

図 1. 価値提案キャンバス

最後に API チームは、API がユーザープロファイルに適合していることをはっきり表現する、いくつかの説明文を記述する必要があります。適合性の説明を要約して 1 つの包括的な説明文に凝縮させれば、それが API の価値提案となります。適合性の説明文によって API とユーザーの適合がさらに強化され、価値提案が洗練されます。メッセージング API を例にとると、説明文は以下のようになります。

> 「顧客からの声をここに記載します。サイドバーコンテンツを追加するときは、言語の違いによる文字数の増加に注意してください。ドイツ語などの一部の言語では、英語よりも広いスペースが必要です。翻訳に対応するには、サイドバーの 1/4 以上を空けておいてください」

「たかが内部向け API を作るだけなのにこれはやり過ぎだ」と思うかもしれません。これは自然な反応ですが、価値を重視する姿勢は、内部ユースケースでも極めて大切です。価値提案の検討が不十分なままでは、API の価値を他のチームに啓発して売り込むのに多大な時間がかかってしまいます。価値提案がきちんと定義されていれば、API プログラムはビジネスに大きく貢献することができます。

[1] Strategyzer、「Value proposition canvas」、2023 年 9 月にアクセス。

API プログラムの価値に対する検討事項

API プログラムの価値を定義するには、以下の 3 つの領域を検討します。

1. **ユーザーの特定：**関係 (顧客、パートナー、開発者)、役割、優先度に基づいてユーザーを特定します。
2. **ペインとゲインへの対処：**図 1 の価値提案キャンバスを参照して、ユーザーのペイン、ゲイン、重要なニーズを判断します。メトリクスの改善 (速度、収益、コスト) と新しい機会の可能性を測定します。
3. **サポートされるユースケース：**価値提案キャンバスを使用して、効果的なペインリリーバーとゲインクリエイターを特定します。これらの特定のユースケースに対応するように API を設計します。
4. **将来的な価値の拡張：**価値提案は、将来を見据えて計画します。継続的に価値を拡大するため、今後のマイルストーン、動向、技術面のイノベーションを予測します。
5. **内部の組織的価値：**API の内部的なメリットと他のチームへの潜在的な価値を評価します。

これらの質問に答えることで、API プログラムの明確な価値提案を確立し、ユーザーのニーズおよび組織の目標に沿ったものとすることができます。

ベストプラクティス #2：最初からビジネスモデルを明確化する

成功する API を作成するには、価値提案だけでは不十分です。API を適切に定義されたビジネスモデルと組み合わせる必要があります。API の価値の認識と伝達は重要ですが、そのコストを財務面のメリットや具体的なメリットとバランスを取る必要もあります。自費出版の書籍「Business Model Generation」において、Alex Osterwalder 氏は組織のビジネスモデルを「組織が価値を提案、作成、提供、獲得する方法」と定義しています。

API のビジネスモデルを理解する

ビジネスモデルキャンバスは、以下のようなビジネスモデルのコアコンポーネントに分解されます。[2]

1. **価値提案：**独自の API 価値を定義します。
2. **収益ストリーム：**API がどのように収益を生成するかを概説します。
3. **コスト構造：**API のメンテナンスに関連するコスト。
4. **顧客セグメント：**ターゲットとするユーザーグループ。
5. **顧客関係：**組織がユーザーと対応する方法。
6. **チャネル：**ユーザーにリーチするための配信手段。
7. **主要パートナー：**API を成功させるための重要なコラボレーション。
8. **主要アクティビティ：**API 運用に必要な基本タスク。
9. **主要リソース：**API 実装に必要な重要アセット。

図 2. ビジネスモデルキャンバス

API は、既存のアセットを使用して新たな可能性を生み出せます。しかし、API の価値を認識することは重要ですが、そのコストを管理することも欠かせません。モデルが適切でない場合、API に価値があってもコストが上昇してしまいます。

API のビジネスモデルに対する検討事項

ビジネスモデルと API の使用方法を一致させるには、以下の 5 つの重要な領域を検討してください。

1. **組織に対する API の価値：**API が組織によるリーチやイノベーションの拡大を支援する方法を確認して、金銭面以外の多様な価値を評価します。
2. **価値の獲得：**特定した価値を獲得する最適な方法を探り、最大の価値を実現するために障壁を最小化します。
3. **コストの対処：**API 関連のコストを認識します。このコストは、エンジニアリングやマーケティングなど、API チーム以外から生じることも少なくありません。これの対処方法を計画します。
4. **長期的なコミットメント：**API には、初期投資の他に、運用およびメンテナンスの継続的な取り組みが必要であることを理解します。
5. **戦略的パートナーシップ：**必須のパートナーシップを特定し、API 開発や市場参入の際にパートナーやサプライヤーの補完的なオファリングを使用します。

ベストプラクティス #3：ユーザーを考慮した設計と実装

API 設計には、一貫したユーザーエクスペリエンスのための基本的な設計原則があります。熟練した開発者が直感的に API を理解できる、「すぐに使える」設計状態を達成することを目標とします。API 設計では、シンプルさと柔軟性という、2 つのコア領域に注目する必要があります。

シンプルさ

API 設計におけるシンプルさはコンテキストに依存し、ユースケースによって設計の複雑性が異なります。API メソッドの粒度の最適なバランスを取ることが重要です。シンプルさはさまざまな面で検討できます。

1. **データ形式：**XML、JSON、プロプライエタリーな形式、またはこの組み合わせ。
2. **メソッド構造：**メソッドはジェネリックなものから高度に特化されたものまでさまざまで、一般には特定のユースケースを達成するためのシーケンスになっています。
3. **データモデル：**公開された API データモデルは基盤のモデルとは異なる場合があり、使用やメンテナンスのしやすさに影響します。
4. **認証：**認証メカニズムにはそれぞれに異なる強みや弱点があり、コンテキストに応じて選択できます。
5. **使用ポリシー：**開発者の権限とクォータは、分かりやすく、管理しやすくしなければなりません。

柔軟性

シンプルさと柔軟性のバランスは重要です。API がシンプルすぎると、特定のユースケースにしか対応できず、適応性が制限されてしまいます。柔軟性を確立するには、基盤システムやデータモデルなど、潜在的な運用事例の概要を規定します。これらの運用の中から、実現可能で価値のあるサブセットを定義します。シンプルさと柔軟性のバランスを適切に調整するには、以下のことを検討します。

1. **アトミック操作を公開する：**アトミック操作を組み合わせて、運用事例の全体をカバーできるようにします。
2. **一般的な、価値のあるユースケースを特定する：**上記とは別に、アトミック操作を組み合わせてこれらのユースケースに対応する、メタ操作のレイヤーを設計します。

API プロトコル設計の検討事項

API 開発において REST アーキテクチャスタイルが優勢な標準であることは依然として変わりませんが、API プロトコルはさらに多様化が進んでいます。近年のコンセプトとして、ストリーミング API や WebSocket があります。しかし、従来型の Web サービス API も消えるわけではありません。プロトコルを選択する際には、ユーザーにとって最も関連性の高いものを選び、次にシンプルさと柔軟性の適切なバランスを取るという原則に従ってください。REST API を提供するインフラストラクチャは進化し続けています。

API 設計を熟考する際には、以下の領域を検討してください。

1. **ユースケースとの一致：**API は特定したユースケースをサポートするように設計し、革新的で頻度の少ないシナリオにも対応できるよう柔軟性を維持します。
2. **RESTful の入念な検討：**RESTful API は最新の技術ですが、本当にニーズに適しているかを確認します。別のアーキテクチャスタイルのほうが、特定のユースケースには適していることもあります。
3. **データモデルの抽象化：**API の実装ではデータモデルとの間に抽象化レイヤーを挟むようにし、どうしても必要な場合を除いてデータベースには直接アクセスしないようにします。
4. **地理的な考慮事項：**データセンターを主なユーザーのいる地域の近くに戦略的に配置して、レイテンシーや可用性など、機能以外の側面を考慮します。
5. **製品との統合：**構造の明確な内部および外部のコミュニケーションを維持しながら調整や分離することで、API 設計をその他の製品と調和させます。

API 実装とデプロイのインフラストラクチャ

組織がデータおよびアプリケーションのインフラストラクチャをクラウドやハイブリッドクラウドに移行するのに伴い、API 実装とデプロイメントのインフラストラクチャもさらに多様化しています。API をマイクロサービスに基づいて構築し、モノリシックに構築された API とマッシュアップさせることができます。API は、コンテナ、仮想マシン (VM)、ベアメタル、またはパブリッククラウド環境内で動作させることができます。

他のあらゆるアプリケーションの場合と同様、自動化された継続的インテグレーション/継続的デリバリー (CI/CD) パイプラインは API ライフサイクル管理にとって重要です。Argo CD および GitOps を導入すると、一元化された API 構成管理、自動化されたデプロイメント CD を提供でき、API チーム間の共同作業が促進され、迅速な開発や API 製品品質の向上が実現します。

ベストプラクティス #4：API 運用を最優先にする

API を公開したら、API プラットフォームチームはこの API を管理し、開発者の想定どおりにアクセス可能で提供されるようにします。ベンダーはソリューションを提供しますが、成功するためには適切な戦略を選択することが不可欠です。API 管理には、以下の 2 つの主要な機能があります。

1. 内部プロセスを最適化して効率化し、コストを削減する
2. 運用を効率的にして、外部開発者がプログラムに対して抱いている期待に対応する

「優れた API の構築、パート 1: ゴールドスタンダード」および図 3 の API 運用のドーナツ図を参照して、これらの目標の達成に役立ててください。

API 運用のドーナツ図

API 運用のドーナツ図を使用して、組織の API 戦略を達成するための運用戦術を定義できます。ドーナツの内側の円は組織の内部活動と効果を示し、円の外部は外部の効果を示します。

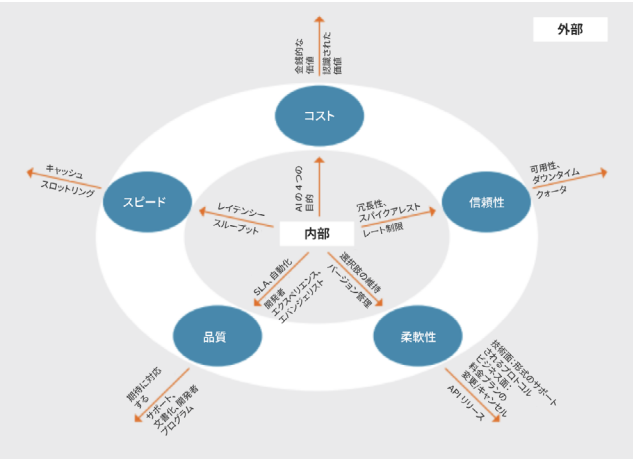

図 3. API 運用のドーナツ図

1. **信頼性：**冗長性やクォータおよびレート制限により、API の可用性を高めます。これらはビジネスモデルと適合し、ダウンタイムを防止します。
2. **柔軟性：**料金プランの変更やキャンセルなど、技術面やビジネス面の導入オプションを提供します。柔軟性を高めると、内部の作業やコストが増える可能性があることに留意してください。
3. **品質：**サービスレベル契約 (SLA) および効率化したプロセスを使用して、開発者の期待に常に応えられるようにします。
4. **速度：**スロットリングやキャッシュなど、ビジネスモデルと適合された技術を使って、低レイテンシーと高スループットを実現します。
5. **コスト：**開発者の価値を最適化し、品質を損なうことなく内部コストを最小化します。

運用に関するこれらの課題の多くに対応するための技術的インフラストラクチャを提供しているベンダーは、Red Hat を含め複数あります。ベンダーを使用することでこれらの問題をコスト効率よく解決できることが多いですが、綿密な戦略が必要です。

API 管理の基本コンポーネント

API エコシステムの運用には、管理を効率的に行うための一連のコンポーネントが必要です。これらのコンポーネントは API 戦略によって異なりますが、コアとなる 3 つのコンポーネントは同じです。

1. **アクセス制御：**認証および承認システムを実装し、アクセスを許可して受信トラフィックを識別します。
2. **レート制限と使用方法のポリシー：**トラフィックにクォータと制限を適用して、負荷を予測します。
3. **分析：**トラフィックパターンを獲得して分析し、API の使用を監視します。

API 運用の検討事項

API 運用戦略を組織の全体的な戦略と調和させると、API の重要度に従ってリソースを割り当てやすくなります。

API 運用計画を熟考する際には、以下の領域を検討してください。

1. **アクセス制御：**誰がアクセスし、アクションを実行し、制限を提供するかを定義して、セキュリティ重視の API 使用方法を維持します。
2. **メトリクスとアラート：**分析によって状況を把握し、カスタマイズされたメトリクスを測定し、API パフォーマンスのアラートを設定します。
3. **スパイクの管理：**アクセス制御とポリシーを使用してインフラストラクチャを計画し、スパイクアレストやスロットリングなどのフォールバックメカニズムを確立します。
4. **API アップタイムの責任：**API アップタイムの責任所在を明確にしておくことは、価値の創成や獲得に直接結び付くため、重要です。
5. **望ましくない使用への対処：**想定された使用と、想定外の望ましくない使用とを区別し、プロアクティブな運用と利用規約によって管理します。
6. **コミュニケーション：**内外の開発者に対する明確なコミュニケーション計画を用意し、計画されたダウンタイム、メンテナンス、API 変更を伝達します。

ベストプラクティス #5：開発者エクスペリエンスについて考慮する

開発者エクスペリエンスは API 設計と似ているように思われますが、それを大きく超えるものです。開発者エクスペリエンスとは、API 自体ではなく、API のパッケージ化および提供と考えてください。すばらしい設計の REST やコンテナ化された API があるとしても、アクセスするためのサインアップ、ドキュメントの確認、およびテストが難しければ、その開発者エクスペリエンスは粗末であり、API の成功に深刻な影響を与えかねません。

API の成功は優れた設計だけでは不十分

シンプルで柔軟な設計の API でも、開発者が興味を持って最終的に導入しなければ無駄になります。同時に、入念に検討された API 設計は開発者エクスペリエンスと導入に大きな影響を与えます。導入は、開発者エクスペリエンスの重要な部分です。

API および API 管理の分野の先駆者の 1 人である John Musser 氏は、興味を高めるためのいくつかの方法を提案しており、それは今でもそのまま通用します。

1. API の目的を明確に定義する
2. サインアップを容易にし、自由にアクセスできるようにする
3. 価格設定に透明性を持たせる
4. 徹底的に文書化する

API 設計を改善して導入を容易にするための主なメトリクスは Time To First Hello World (TTFHW) で、API の初期受け入れを測定するものです。また、開発者プログラムを重視する Time to first profitable app (TTFPA) では、より広範なコンテキストが測定されます。収益性は API およびビジネス戦略に応じて定義が異なるので、このメトリクスの扱いには少し注意が必要です。とはいえ、API プログラムの一部としての API 処理に関する事項を考えることができるので、TTFPA は検討に値します。

開発者エクスペリエンスには、価値のある製品/サービス設計と、容易なアクセス性という、2 つの原則があります。ポータル、コミュニティ、エバンジェリスト、イベント、測定など、堅実な開発者向け取り組みプログラムにより、取り組みが強化されます。開発者ポータルの代表的なメトリクスの例として、ページ訪問数、サインアップ数、API トラフィック、サポートリクエスト数などが挙げられます。イベントは、参加者、ハッカソンでの API 導入、リードの数で測定できます。「イベントでの講演によって API のサインアップが増えたか?」など、コリレーションを考えることも役に立ちます。

開発者プログラムには、以下の要素を含めます。

- **コミュニティ構築：**ハッカソンなどのイベントにより、交流を深め、導入を促進します。
- **開発者エバンジェリスト：**API のメリットを宣伝し、関心を持たせる、成功するためには欠かせない存在です。
- **パイロットパートナーと事例：**アーリーアダプターからはフィードバックと事例を得ることができます。
- **エコシステムパートナー：**パートナーシップによる相乗効果で、導入を増加させます。
- **測定：**メトリクスを API の目的と適合させ、効果的な管理に役立てます。
- **コミュニケーション：**ポータル、ニュースレター、Slack または Discord のプライベートサーバーなど。

対象ユーザー向けに調整された開発者プログラムを作成すると、取り組みやエクスペリエンスに役立ちます。

開発者エクスペリエンスの評価の検討事項

十分に練られた開発者エクスペリエンスを提供すると、開発者はその潜在能力を最大化できるようになります。開発者エクスペリエンスを評価して、単なるコードの羅列ではなく、創造性と生産性を導くゲートウェイとなる API の構築に役立てるための、6 つの基本的な検討事項を紹介します。

1. **価値の説明：**簡単なエレベーターピッチを作成して、API の価値を開発者に伝達します。
2. **TTFHW と TTFPA：**すべての開発者エクスペリエンスの要素 (ポータルなど) を考慮し、Time to first hello world と Time to first profitable application を評価して最小化します。
3. **オンボーディングプロセス：**オンボーディングを API ユースケースと一致させ、開発者が早期段階で成功できるようにシンプルさを維持します。
4. **価値の準備：**API が十分な価値を提供して、開発者を引きつけて維持できるようにします。
5. **開発者サポート：**ドキュメント、FAQ、フォーラムによるセルフサービスサポートを優先し、重大度の高い問題用に追加のメカニズムを用意します。
6. **想定とは異なる使用：**標準的ではないユースケースを探求する開発者向けのサポートとドキュメントに対応します。

ベストプラクティス #6：マーケティングの基本を越える

API を開発者に売り込むのは、特に提示される価値が開発者のビジネスまたは技術のニーズと一致しない場合、困難です。API はその他の製品と同様にして売り込み、セグメンテーション、ターゲティング、ポジショニング (STP) に一致させます。効果的なマーケティングは、API の価値と STP の原則に従って、適切な API を適切な開発者とマッチングさせます。

1. **セグメンテーション：**顧客を、内部ユーザー、パートナー、エンドユーザー、外部開発者に分類します。効果的なエンゲージメントを行うため、ジョブ理論の手法を使用して、広大な開発者市場をセグメント化します。
2. **ターゲティング：**アクセス性、実質性、差別化に基づいて、セグメントの魅力を評価します。有望なセグメントを選択し、それに従ってマーケティング戦術を調整します。
3. **ポジショニング：**特定のニーズに対処し、ペインを解消し、選択した開発者セグメントにゲインを提供することで、API を目立たせます。

マーケティングにおいて開発者エクスペリエンスを優先することは、成功するために重要です。開発者エバンジェリスト、堅牢な開発者ポータル、ハッカソン、その他のイベントなどの戦略により、開発者との強固な関係を作ることができます。

API のマーケティングの検討事項

1. **対象者：**主なユーザーグループを優先し、内部ユーザー、パートナー、顧客、一般社会に基づいて段階を踏んで API を進化させます。
2. **スペシャリストの選択：**エンジニアリング、サポート、営業、製品管理などの分野を考慮して、API の価値提案と一致するエキスパートを選択し、効果的なプロモーションを行います。
3. **イベント戦略：**API の目標に応じてイベントの種類 (水平または垂直、グローバルまたはローカル、カンファレンスまたはハッカソン) を選択します。
4. **ハッカソンの適合性：**ハッカソンの関連性 (サインアップ、SDK ダウンロード、アプリケーション、リクルート、ブランディング) を評価し、それに応じて計画します。
5. **内部マーケティング：**部署を越えて支援を取り付け、マーケティング部門と相談して明確化し、製品チームと顧客にメリットを伝達します。

ベストプラクティス #7：API の廃棄と変更管理を覚えておく

API へのアドバイスは、API の設計、作成、運用に偏りがちです。しかし、API のライフサイクルの中で見過ごされている最も重要な要素の 1 つは、導入して運用を開始してから数カ月後に発生しています。つまり、API の廃止を含む、API の更新の管理です。

突然変更したことが原因となって障害を発生させると信頼が損なわれ、特に開発者を把握できていない場合、モバイルアプリケーションの承認が絡む場合、およびデバイスにアップデート機能がない場合には、多大なコストが生じます。

API の変更は通常、非破壊的なものと破壊的なものに分類されます。非破壊的変更には新しいメソッドや機能強化があり、破壊的変更には削除、修正、全面的な廃止があります。メジャーバージョン番号と移行計画では破壊的変更に対応し、マイナーバージョンは非破壊的変更に対応します。変更によってどのアプリケーションも破壊されないようにすることは困難なので、API に何らかの変更を行う場合、非破壊的であるとしても、以下の方法でロールアウトすることを強く推奨します。

- 導入開始前に、新しいバージョンのテストエンドポイントを提供する
- 開発者に E メールまたはその他の通知で変更について知らせ、いつ、どのような変更を行うかを伝える

効果的なコミュニケーションと透明性のある契約が重要です。問題の詳細を伝達し、コミットメントを支持し、バージョンのサポート期間を示します。たとえば、API の廃止には、通知の期間延長、報道機関への対処、移行計画、必要に応じてデータエクスポートツールなど、構造化されたアプローチが必要です。

移行計画によって API のアップデートが容易になります。

1. テスト用に新しいバージョンを導入する
2. 古いバージョンの廃止をユーザーに通知する
3. ユーザーに移行支援を提供
4. 古いバージョンを廃止する

包括的な API 管理には、アップデートと廃止の予想、効果的なコミュニケーション、透明性のある行動による信頼の維持が含まれます。

API のライフサイクルの検討事項

1. **顧客の保証に向けたコミットメントの維持：**これはサービスのどの程度の安定性をユーザーに保証するかということで、おそらく API プログラムにとって最も重要です。サービスの安定性に対するユーザーへのコミットメントを定義します。このコミットメントは導入に影響を与えます。
2. **変更および破壊的変更のプロセス：**保証を維持するためのリリース手順を決定します。関連する当事者と承認ステップを特定します。
3. **変更の伝達：**API 定義形式を使用して、変更を検出、文書化、伝達します。互換性を維持し、明確なコミュニケーションを行います。
4. **バージョン管理：**開発者とユーザーの ID を使用して、以前のバージョンの使用状況を監視します。問題を防ぐため、廃止プロセスを確立します。
5. **製品の一致：**顧客対応を考慮しながら API の進化を関連する製品の変更と調整し、必要な適合に対処します。

API 戦略の維持

これらのベストプラクティスは、API 戦略の提議、実装、強化を支援するためのものです。戦略を適応させて、新しい API のチャンスを見いだすことが目標です。おそらくはこれまでのセクションの質問の多くに詳しく回答する必要があり、新たなチャンスやリスクがそのうち生じてくる可能性があります。以下に例を示します。

- API の価値を顧客に拡張する方法はあるか?
- 将来に向けて十分な準備ができているか?
- 開発者が心から取り組みたいと思うほど価値提案に魅力があるか?

IT を取り巻く環境が進化するにつれて、API とオファリングも進化する必要があります。変化には、以下の 4 つの主な原動力があります。

1. **業界の力:** 新しい競合他社やサービスによって API が置き換えられる可能性があります。
2. **市場の力:** ユーザーの要求や市場セグメントの状況が変化します。
3. **マクロ経済の力:** グローバル市場の変化によってユーザーの予算が影響を受けます。
4. **動向:** テクノロジーや規制要件が進化します。

永続的な API プログラムの作成

潜在顧客を持つ強力なユースケースに一致する API プログラムには、導入開始、成長、進化を成功させるポテンシャルがあります。このような成功する API プログラムでは、API の計画的な柔軟性によってイノベーションを起こす余地を作り、設計と運用を通じて潜在的な負の使用シナリオに事前に対処します。API の利用規約に、予期しない振る舞いに対して行動を起こすために必要な内容を盛り込んでおくこともまた、重要です。

以下の場合、API 戦略を再評価する必要がある可能性があります。

- 予期しないイノベーションを利用しているために価値が損なわれている場合。このアプローチとともに、堅牢なインフラストラクチャも成功のためには重要です。
- 自社やユーザーに対して説得力のあるユースケースがなく、別のアプローチが必要とされる場合。
- 否定的な振る舞いに対する疑いが内部で払拭できず、対策やコミュニケーションが不十分であることが示される場合。
- API の侵害や悪用が頻発し、意図した価値と実際の価値との乖離が示される場合。

API は何もないところに存在するのではない

テクノロジーを取り巻く状況は常に進化しており、さまざまな強力なクラウドネイティブ・アプリケーション開発ツールが導入されました。Kubernetes、Red Hat® OpenShift®、Red Hat OpenShift Service Mesh などがそれに該当し、相乗効果によって接続性を強化し、卓越したアプリケーションの開発を促進します。しかし、このような進化の中であっても、堅牢な API 戦略の基本原則と効果的な API 管理システムの重要性に変わりはありません。この取り組みに着手したら、全体的な戦略に一致するだけでなく、これらのプラットフォームの確固とした理解または一貫した統合を実証し、戦略上のビジョンの履行を促進する API 管理ソリューションを模索しましょう。

API の構築と管理に Red Hat を選ぶ理由

Red Hat は戦略的な API 設計ファーストのアプローチを支持し、API プログラムの成功を支援します。このアプローチは、リソースマッピングやビジネス・シナリオ・モデリングを含む初期計画と設計から、アクセス制御、API のアクセス、使用状況の分析を含無 API の管理まで、API のライフサイクル全体を包含します。

この API オーナーのマニュアルで説明してきた API 設計ファーストのアプローチのベストプラクティスで開発チームを支援するため、Red Hat では、Red Hat 3scale API Management を提供しています。これはハイブリッドクラウド環境専用に作成された API 管理ソリューションです。Red Hat の分散型で軽量なコンテナベースのクラウドネイティブ・ソリューションにより、大規模なワークロードを処理できるようになり、セキュリティ重視のアプローチとコラボレーションが促進されます。パブリッククラウドおよびプライベートクラウド環境上でのサービス、リソース、アプリケーション、エンタープライズシステムへのアクセスを共有および制御して、複雑なエコシステムにおけるビジネスを強化します。

しかし前述したように、API は何もないところに存在するのではなく、これらのベストプラクティスのいくつかを使用するには、クラウドネイティブ・アプリケーション開発向けの完全なアプリケーション・プラットフォームが必要になります。このため、Red Hat 3scale API Management は Red Hat Application Foundations ポートフォリオに含まれています。これは、統合、API 設計、API ガバナンスとレジストリ、Red Hat OpenShift Container Platform と連携するために設計されたストリーミングおよびメッセージングなどの機能を含む、ミドルウェアツールのコレクションです。Red Hat OpenShift と Red Hat Application Foundations を併用することで、完全なクラウドネイティブ・アプリケーション・プラットフォームが実現し、新しいソフトウェアをユーザーにより効率的かつ安全に提供し、組織は戦略的な機能とメリットを手に入れることもできます。

クラウドネイティブ・アプリケーション開発アプローチを Red Hat 3scale API Management、Red Hat Application Foundations、Red Hat OpenShift で向上させ、最適化された効率的で協調的なアプリケーション構築の未来を体験しましょう。

まとめ

これらのベストプラクティスが、これからする選択の指針として、少なくとも投げかけられた質問についてお役に立てれば幸いです。API 戦略を構築するプロセスは、API の価値を明確で一貫した方法で組織全体で理解することから始まります。しかし、この API に対する価値やビジネス目的は固定されたものではありません。物事は変化し、API 戦略はそれとともに変化する必要があります。

Red Hat はお客様とともに取り組み、API エコノミーで観察してきました。その経験から、重要な API プログラムの特性のいくつかをまとめることができます。

- ユーザーが API を導入するのは、そのユーザーにとって価値があるからです。API は、ペインの解消とゲインの創出により、基本的な役割を果たします。
- API はユーザーに価値を提供し続け、環境に応じて変化する価値提案と戦略を提供します。
- API は組織に対して内部的に価値があります。重要なことを実行し、組織の重要な目的の達成を支援します。
- 可能な限り多くの関係者 (理想的には全員) が満足します。

Red Hat 3scale for API Management の詳細と、Red Hat の API 管理リソースをご確認ください。