Red Hat

API 使用手册

成功 API 团队七大最佳实践

Manfred Bortenschlager

红帽基于 API 的集成解决方案和
API 管理业务开发总监

Andrew Mackenzie

红帽软件工程 API 管理总监

Jaya Baskaran

红帽首席技术营销经理

Greg Pack

红帽应用服务和 API 管理解决方案
产品营销高级经理

Red Hat

目录

内容摘要 3

简介 3

我们为何需要实施 API? 4

我们希望谁使用这些 API? 4
我们希望通过这些 API 实现哪些具体成果? 4
我们如何计划执行 API 程序来实现这一目标? 5
API 团队 5

最佳实践 1：始终关注 API 的价值 6

用 API 术语来讲，这意味着什么? 6
API 程序价值的注意事项 7

最佳实践 2：从一开始就明确业务模式 7

了解 API 业务模式? 7
API 业务模式的注意事项 8

最佳实践 3：在设计和实施时考虑用户 9

简单性 9
灵活性 9
API 协议设计注意事项 9
与产品集成 9
API 实施和部署基础架构 10

最佳实践 4：将 API 运维放在优先位置 10

API 运维圆环 10
API 基本管理组件 11
API 运维注意事项 11

最佳实践 5：关注开发人员体验 12

API 的成功需要的不只是好设计 12
评估开发人员体验的注意事项 13

最佳实践 6：超越营销基础知识 13

营销 API 的注意事项 13

最佳实践 7：不要忽视 API 停用和变更管理 14

API 使用寿命注意事项 14
维持 API 战略 15
创建持久的 API 程序 15
API 需要切合实际 15

为什么选择红帽来构建和管理 API? 16

结论 16

内容摘要

现代应用开发、连接和基础架构（如 5G 和边缘计算）的进步继续重塑业务流程的发生方式和地点。虽然这些进步正在提高业务速度，但如果没有现代应用编程接口（API），它们的优势也无法实现。

今天，API 已经成为现代企业的数字纽带，可以为其运维和产品以及合作伙伴战略等各方面增加新功能。无论您已有 API 战略，还是准备推出战略，都可以阅读我们这本电子书，了解成功的 API 程序的七大最佳实践。

简介

阅读本电子书之前，请思考一下您的 API 程序的关键目标。稍后提出的一些问题可能有助于改变、验证您的目标或为您的目标提供更多实质内容。

有效的 API 程序应该建立在企业的总体战略基础之上，并助力其实现目标。您能够回答以下四个问题时，您便可确定自己将会制定出有效的战略：

1. 我们为何需要实施 API?
2. 我们希望谁使用这些 API?
3. 我们希望通过这些 API 实现哪些具体成果?
4. 我们如何计划执行 API 程序来实现这一目标?

我们为何需要实施 API?

考虑 API 时，我们一定要全面了解其价值和目的。避免常见的误解对于最大限度地发挥其潜在的业务价值至关重要。一个普遍的误解是，只有当用户付费时，API 才有价值，而当 API 作为产品时，情况确实如此。然而，许多 API 程序仍然是内部项目，这就会有不同的指标，如上市速度、可重用性、销售、品牌知名度或联盟推荐。

五种典型的 API 提供商用例包括：

1. **扩展全渠道：**通过移动应用、物联网等多种渠道扩大可访问性。
2. **不断发展您的生态系统：**培养客户或合作伙伴生态系统以进行扩张。
3. **扩大您的影响力：**建立广泛的交易或内容分发网络。
4. **推动新的业务模式：**通过新颖的业务模式推动创新。
5. **产生内部创新：**在企业内部产生创新。

在现代科技公司中可以找到许多成功的 API 战略示例，包括亚马逊、Salesforce、Twilio 和其他公司。但这并不意味着只有这些领先的科技公司才能使用 API 实现盈利、创新、合作伙伴关系增长或敏捷性。许多成熟的企业都有很多传统系统，这些系统并不会突然消失（甚至可能永远不会消失）。API 是与这些传统系统集成、在内部或外部公开这些系统并将它们与其他服务混合以创造新价值的完美粘合剂。任何正在进行数字化转型的企业都可以实施结构良好的 API 战略，从而利用 API 经济的优势。

这个优势必须足够诱人，以吸引企业明确选择决定投资 API。

我们希望谁使用这些 API?

了解您的 API 最终用户将有助于定义您的 API 程序成功指标。

- **内部最终用户：**内部团队可以使用 API 来访问公司信息，用于任意数量的业务目标。
- **外部最终用户：**外部第三方开发人员可以在其应用中使用相同的 API 或子集。

这个优势必须足够诱人，以吸引企业明确选择决定投资 API。

我们希望通过这些 API 实现哪些具体成果?

要确定 API 驱动的具体结果，请从企业内部和外部角度考虑：

- **内部：**使用独特且有价值的资产。拥有独特数据的企业，如 Meta 的“点赞”数据，可以提供有价值的 API。
- **外部：**考虑市场动态、趋势、竞争对手和客户行为。外部力量塑造业务战略并影响 API 功能。

市场动态深刻影响映射 API。早期的映射公司忽视了帮助 Waze 这样的初创公司蓬勃发展的实时需求。谷歌收购了 Waze，将其技术集成到一个成功的 API 中。Twitter、Reddit 和其他巨头也接受了用于创新和生态系统增长的 API。

正确的 API 战略经常将内部资产与外部市场洞察力相结合。

我们如何计划执行 API 程序来实现这一目标?

最后一个问题，“我们如何设计 API 程序来实现我们的需求？”全在于实现和执行，这需要完善设计：

1. **技术选择：**确定构建 API 的技术堆栈。
2. **设计和维护：**制定 API 设计和维护战略。
3. **推广和营销：**在企业内推广 API，并在外部进行营销。
4. **资源分配：**为 API 的开发和维护分配必要的资源。
5. **团队组成：**组建一支有能力的 API 开发和管理团队。
6. **开发人员社区：**为外部和内部开发人员创建和维护专门的沟通计划。
7. **成功指标：**建立方法，根据业务目标跟踪成功情况。

通过揭示 API 的“原因”、“适用人群”、“内容”和“方式”，企业可以在不断发展的 API 经济中推动创新和增长。对于这些问题，每个企业的回答都会有所不同，并且会受到您的目标、部署的战略以及至关重要的 API 团队的影响。

API 团队

API 团队的结构通常与其他产品团队类似。无论您的客户是内部客户还是外部客户，API 团队都负责构建、部署、运维和优化其他人所依赖的基础架构。

就像产品团队一样，API 团队也可以多样化，团队应该包括以产品为中心的人员（战略和目标负责人）、以设计为中心的团队成员（确保最佳 API 设计实践）、能将 API 技术变成代码的工程师，以及最终运行 API 的运维团队成员。随着时间的推移，您可能还需要其他人员加入，包括支持和社区团队成员、API 专家和安全代表。您的扩展 API 团队还可以包括您的开发人员社区的成员。

虽然这个团队可能人数众多，但有些人可能身兼数职，尤其是在较小的企业中。重要的是要确保能够代表所有利益相关者的意见，甚至可以让一位团队成员专门负责解决他们的顾虑。

在许多情况下，API 团队是临时组成的，可能属于不同的企业单位，有不同的直属经理。这种结构会导致定义 API 的共同愿景变得特别具有挑战性。对于大型 API 程序来说，不同的 API 团队需要展开协作。

无论您的企业规模如何，本电子书中描述的七大最佳实践都将能够帮助您建立一个成功的 API 团队，并且最终可能会容纳更多的人。

最佳实践 1：始终关注 API 的价值

API 程序必须优先考虑提供价值的核心目标，同时避免复杂性。价值主张决定了用户效用，这是 API 成功的关键驱动因素。一个有吸引力的价值主张对于有效的营销至关重要。将其与公司目标协调一致可确保可持续性，使成熟公司能够通过 API 增强其产品。

Alex Osterwalder 的 API 价值模式将用户利益与 API 功能相结合，在用户需求与 API 价值之间建立关键的“适配”。[1]解决需求、缓解痛点和创造价值至关重要。

用 API 术语来讲，这意味着什么？

在这个迭代过程中，第一步描述了用户试图完成的工作，如在紧急情况下自动发送紧急通信、备份关键文件以及采样数据以检测某些事件。

第二步需要确定在尝试完成工作之前、期间或之后影响用户的具体痛点。其中包括确保可靠的多尝试消息发送、故障检测、管理多条消息、基于位置的消息系统集成、用最少的带宽保护文件传递，以及广泛数据量的实时相关性。

构建用户配置文件的第三步包括概述潜在的好处，如其他类型的通知，这些通知可以创造机会而不是警告威胁，如果可靠性足够好，则无需其他存储设备，并根据事件自动触发操作。

转向价值映射，应该映射出 API 的功能、特性和服务，重点关注解决方案和创收点。该过程形成了具体的示例，如确保消息传递的消息传递 API、用于高效版本更新的存储同步 API，以及提供可配置数据流的数据聚合 API。

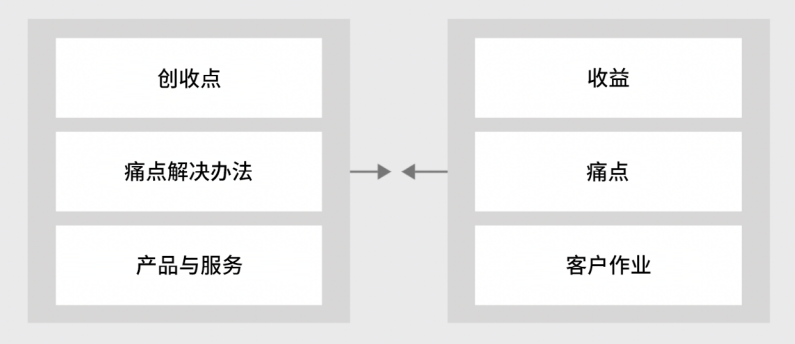

图 1. 价值主张画布

最后，API 团队应完成一个说明练习，编写几个语句，展示 API 和用户配置文件之间的适配。然后，当您将适配语句压缩为一个总体消息时，它就成为您 API 的价值主张。这些“适配”语句进一步巩固了 API 与用户的一致性，提炼出了价值主张。对于消息传递 API 示例，该语句可能是这样的：

> *“此处显示客户的报价。添加侧边栏内容时要注意语言扩展。德语等部分语言需要的空间比英语多。为了确保译文空间合适，至少将 1/4 的侧边栏留空。”*

您可能会觉得“这似乎完全矫枉过正了，我们的 API 只是一个内部 API。” 这种反应很正常，但即使在内部用例中，对价值的关注也至关重要。定义不明确的价值主张会导致需要花费大量时间将 API 推销给其他团队并提供培训。定义明确的价值主张则使 API 程序成为业务的主要促进因素。

[1] Strategyzer，“价值主张画布。”2023 年 9 月。

API 程序价值的注意事项

为定义您的 API 程序的价值，请考虑以下关键领域：

1. **用户识别：**根据用户的关系（客户、合作伙伴和开发人员）、角色和偏好来识别用户。
2. **解决痛点和收益：**参考图 1 中的价值主张画布，将确定用户的痛点、收益和关键需求。衡量指标改进（速度、收入和成本）和新机会的潜力。
3. **支持的用例：**使用价值主张画布来确定有效的解决方案和创收点。设计您的 API 以满足这些特定的用例。
4. **未来的价值扩展：**着眼于未来规划您的价值主张。预测未来的里程碑、趋势或技术创新，以实现持续的价值增长。
5. **内部企业价值：**评估 API 的内部效益和对其他团队的潜在价值。

通过回答这些问题，您的 API 程序可以建立明确的价值主张，并确保与用户需求和企业目标保持一致。

最佳实践 2：从一开始就明确业务模式

创建一个成功的 API 需要的不仅仅是一个价值主张。它需要将 API 与定义良好的业务模式相结合。尽管认识和传达 API 的价值很重要，但其成本也必须与其财务或切实效益相平衡。Alex Osterwalder 在其联合撰写的《业务模式生成》一书中，将企业的业务模式定义为企业提出、创建、交付和获取价值的方式。

了解 API 业务模式

业务模式画布剖析了业务模式的核心组件，其中包括：[2]

1. **价值主张：**定义独特的 API 价值。
2. **收入流：**概述 API 如何生成收入。
3. **成本结构：**维护 API 的相关成本。
4. **客户细分市场：**目标用户群。
5. **客户关系：**企业如何与用户互动。
6. **渠道：**接触用户的分发方法。
7. **关键合作伙伴：**API 成功的关键合作。
8. **关键活动：**API 运维所需的基本任务。
9. **关键资源：**API 实施所需的重要资产。

图 2. 业务模式画布

API 可以利用现有资产打开新机会。尽管认识到 API 的价值至关重要，但其成本管理也不容忽视。虽然 API 有价值，但不合适的模式可能会导致成本上升。

API 业务模式的注意事项

为了使您的业务模式与 API 的使用保持一致，请考虑以下五大关键领域：

1. **企业的 API 价值：**通过探索 API 如何帮助企业扩大影响力或创新，评估不同的价值，而不仅仅是货币价值。
2. **获取价值：**确定获取已识别价值的最佳方式，最大限度地减少实现最大价值的障碍。
3. **覆盖成本：**识别与 API 相关的成本，通常不只是 API 团队，还有工程或营销等团队，并计划其覆盖范围。
4. **长期承诺：**了解 API 在初始投资之外还需要持续运维和维护。
5. **战略合作伙伴关系：** 在 API 开发和进入市场期间，确定必要的合作伙伴关系，并使用合作伙伴和供应商提供的补充产品。

最佳实践 3：在设计和实施时考虑用户

API 设计包含基本设计原则，以实现一致的用户体验。目标是实现“随时可驱动”的设计状态，让资深开发人员能够直观地理解 API。API 设计应关注两个核心领域：简单性和灵活性。

简单性

API 设计中的简单性取决于环境，不同用例的设计复杂性各不相同。在 API 方法粒度上实现适当的平衡至关重要。可以从不同层面来考虑简单性：

1. **数据格式：**在 XML、JSON、专有格式或组合之间进行选择。
2. **方法结构：**方法从通用到高度特定，通常按顺序排列以实现特定的用例。
3. **数据模式：**公开的 API 数据模式可能与底层数据模式不同，从而影响可用性和可维护性。
4. **身份验证：**不同的身份验证机制各有优缺点，具体选择取决于环境。
5. **使用政策：**开发人员的权利和配额应该是可以理解和管理的。

灵活性

平衡简单性和灵活性至关重要。过于简单的 API 可能只能满足特定的用例，这就限制了它的适应性。为了建立灵活性，概述潜在的运维空间，包括底层系统和数据模式。定义这些运维的可行、有价值的子集。为了在简单性和灵活性之间实现适当的平衡，需要考虑以下方面：

1. **公开原子操作：**组合原子操作以覆盖整个操作空间。
2. **确定常见和有价值的用例：**设计第二层元操作，该操作结合多个原子操作来为这些用例提供服务。

API 协议设计注意事项

虽然具象状态传输（REST）架构样式仍然是 API 开发的主导标准，但 API 协议正变得更加多样化。最新的概念包括流 API 或 WebSocket。经典的 Web 服务 API 也不会消失。协议选择应遵循以下原则：选择与用户最相关的协议，然后在简单性和灵活性之间实现适当的平衡。提供 REST API 的基础架构不断发展。

要仔细考虑 API 设计，请考虑以下方面：

1. **与用例保持一致：**设计 API 以支持已确定的用例，为创新和不常用的场景保持灵活性。
2. **全面的 RESTful：**虽然 RESTful API 很先进，但要评估它们是否真正符合您的需求，因为不同的架构样式可能更适合特定的用例。
3. **抽象数据模式：**实施 API，确保 API 和数据模式之间有一个抽象层，除非必要，避免直接访问数据库。
4. **地理注意事项：**考虑非功能方面，如 延迟和可用性，因此战略性地将数据中心放置在您的主要用户区域附近。
5. **与产品集成：**确保 API 设计通过协调或解耦与其他产品相协调，同时保持结构内部和外部的清晰沟通。

API 实施和部署基础架构

随着企业转向数据和应用的云和混合云基础架构，API 实施和部署基础架构也变得更加多样化。API 可以基于微服务构建，并与构建为整体的 API 以及两者之间的一切混合在一起。它们可能位于容器、虚拟机（VM）、裸机或公共云环境中的某个地方。

与任何其他应用一样，拥有自动化的持续集成和持续交付（CI/CD）管道对于 API 生命周期管理至关重要。采用 GitOps 和 Argo CD 可以提供集中式 API 配置管理、自动化部署 CD，并增强整个 API 团队的协作能力，从而实现快速开发和更好的 API 产品质量。

最佳实践 4：将 API 运维放在优先位置

API 平台团队在 API 上线后对其进行管理，以确保它们可访问，并根据开发人员的期望进行交付。虽然供应商提供解决方案，但选择正确的战略对成功至关重要。API 管理涉及两个主要功能：

1. 精简内部流程以提高效率并降低成本。
2. 确保运维有效，以满足外部开发人员对程序的期望。

构建伟大的 API 第一部分：黄金标准和图 3 中的 API 运维圆环可以帮助您实现这些目标。

API 运维圆环

API 运维圆环可用于定义运维战术，以实现企业的 API 战略。圆环的内圈代表一个企业的内部活动和效果，圆环外的一切都是外部效果。

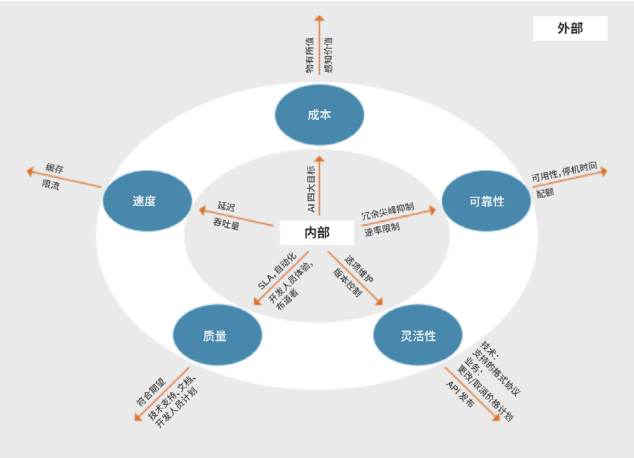

图 3. API 运维圆环

1. **可靠性：**通过冗余或配额和速率限制确保 API 可用性。这些与业务模式保持一致，防止停机。
2. **灵活性：**提供技术和业务采用选项，例如在价格计划或取消之间进行变更。请记住，更大的灵活性意味着更多的内部努力和成本。
3. **质量：**使用服务级别协议（SLA）和简化的流程，一致遵守开发人员的期望。
4. **速度：**通过限流和缓存等技术实现低延迟和高吞吐量，有可能与业务模式保持一致。
5. **成本：**优化开发人员的价值，同时在不影响质量的情况下最大限度地降低内部成本。

需要注意的是，红帽等供应商为其中许多运维挑战提供了技术基础架构。使用供应商通常能够经济高效地解决这些问题，但战略必须全面彻底。

API 基本管理组件

运维 API 生态系统需要一组独特的组件来进行有效管理。虽然这些组件根据 API 战略会有所不同，但核心三个组件是一样的：

1. **访问控制：**实施身份验证和授权系统，以允许访问和识别传入流量。
2. **速率限制和使用政策：**对流量强制执行配额和限制以预测负载。
3. **分析：**捕获和分析流量模式，以监控 API 的使用情况。

API 运维注意事项

确保您的 API 运维战略与企业的总体战略相协调，将有助于您根据 API 的重要性分配资源。

要仔细考虑 API 运维计划，请考虑以下方面：

1. **访问控制：**定义谁访问、执行操作和强制执行限制，确保 API 使用以安全为重点。
2. **指标和警报：**通过分析获得可见性，衡量定制的指标并设置 API 性能警报。
3. **管理高峰：**使用访问控制和战略规划基础架构，并建立高峰抑制或限流等回退机制。
4. **API 正常运行时间责任：**明确 API 正常运行时间的所有权至关重要，因为它与价值产生和获取直接相关。
5. **解决非预期使用情况：**通过主动运维和条款与条件进行管理，区分预期和意外的非预期用途。
6. **沟通：**为内部和外部开发人员制定明确的沟通计划，告知他们计划的停机时间、维护和 API 更改。

最佳实践 5：关注开发人员体验

虽然开发人员体验听起来与 API 的设计相关，但它远远不止于此。将开发人员体验视为 API 打包和交付的一部分，而不是 API 本身。您的 REST 或容器化 API 可能设计非常精美，但如果很难注册访问、读取文档和测试，这就带来了糟糕的开发人员体验，可能会极大地影响 API 的成功。

API 的成功需要的不只是好设计

如果开发人员不使用 API，并且最终也没有采用，那么这款设计简单灵活的 API 就浪费了。同样，全面周到的 API 设计会极大地影响开发人员的体验和采用。采用是开发人员体验的重要组成部分。

作为 API 和 API 管理领域的先锋之一，John Musser 介绍了几种提高参与度的方法，这些方法仍然有效：

1. 明确定义 API 的目标。
2. 注册简单、访问免费。
3. 定价透明。
4. 文档全面。

改进 API 设计以便于采用的关键指标是“首次问世时间”（TTFHW），衡量 API 初始交互。首款盈利应用时间（TTPPA）获取了更广泛的环境，强调开发人员程序。这个指标更为棘手，因为盈利是一个定义问题，具体取决于您的 API 和业务战略。考虑 TTFPA 会很有帮助，因为它会迫使您将 API 运维视为 API 程序的一部分。

开发人员体验涉及两个原则：有价值的产品或服务设计和易于访问。一个稳定的开发人员参与计划可以提高参与度，包括门户网站、社区、宣传者、活动和测量。开发人员门户网站的一些典型指标示例包括页面访问、注册、API 流量或支持请求。活动可以通过参加人数、黑客马拉松 API 的采用率或潜在客户来衡量。创建相关性会有帮助，例如“活动上的演讲是否触发了更多 API 注册？”

开发人员项目应包含以下元素：

- **社区构建：**像黑客马拉松这样的活动促进互动，推动采用。
- **开发人员宣传者：**对成功至关重要的是，他们体验到 API 的优势并进行推广。
- **试点合作伙伴和案例研究：**早期采用者提供反馈和案例研究。
- **生态系统合作伙伴：**合作伙伴关系协同促进采用。
- **测量：**指标与 API 目标一致，有助于有效管理。
- **沟通：**门户网站、时事通讯、私人 Slack 或 Discord 服务器。

为受众量身定制开发人员程序可以提高参与度和体验。

评估开发人员体验的注意事项

提供全面周到的开发人员体验可以激励开发人员最大限度地发挥他们的潜力。以下是评估开发人员体验以帮助构建 API 的六大注意事项，这些 API 不仅是代码行，更是创造力和生产力的门户。

1. **价值解释：**制作一个简短的电梯游说，向开发人员传达 API 的价值。
2. **TTFHW 和 TTFPA：**考虑到所有开发人员体验元素（如门户网站），评估并尽量缩短首次问世时间和首款盈利应用时间。
3. **培训流程：**将培训与 API 用例保持一致，保持简单性以使早期开发人员获得成功。
4. **价值提供：**确保 API 提供足够的价值来吸引和留住开发人员。
5. **开发人员支持：**通过文档、常见问题和论坛确定自助服务支持的优先级，并为更深入的问题提供附加机制。
6. **非常规使用：**为开发人员探索非标准用例提供支持和文档。

最佳实践 6：超越营销基础知识

向开发人员推销 API 可能很困难，尤其是在所提供的价值与开发人员的业务或技术需求不匹配的情况下。API 应该像其他产品一样进行营销，与细分市场、目标和定位（STP）保持一致。有效的营销将合适的 API 与合适的开发人员相匹配，并遵循 API 价值和 STP 原则。

1. **细分市场：**将客户分类为内部用户、合作伙伴、最终用户或外部开发人员。为了有效参与，使用“待完成的工作”方法来细分庞大的开发人员市场。
2. **瞄准目标：**根据可访问性、实质性和差异性评估细分市场的吸引力。选择有前景的细分市场，并相应地调整营销战术。
3. **定位：**通过解决特定需求、缓解痛点并为选定的开发人员细分市场提供收益，让您的 API 脱颖而出。

营销中优先考虑开发人员体验对成功至关重要。开发人员宣传者、强大的开发人员门户网站、黑客马拉松和其他活动等战略可以与开发人员建立牢固的关系。

营销 API 的注意事项

1. **目标受众：**优先考虑关键用户组，根据内部用户、合作伙伴、客户或公众分阶段发展 API。
2. **专家选择：**选择与 API 价值主张一致的专家进行有效推广，同时考虑工程、支持、销售和产品管理等领域。
3. **活动战略：**根据 API 目标选择活动类型（水平或垂直、全球或本地、会议或黑客马拉松）。
4. **黑客马拉松的适用性：**评估黑客马拉松的相关性（注册、SDK 下载、应用、招聘、品牌），并进行相应的计划。
5. **内部营销：**确保跨部门的支持，与营销部澄清，并向产品团队和客户传达利益。

最佳实践 7：不要忽视 API 停用和变更管理

API 的建议主要集中在 API 的设计、创建和运维上。然而，API 之旅中被忽视的最关键部分之一是 API 发布和运维数月后的管理更新，包括 API 的弃用。

突然变化造成的中断可能会削弱信心并带来巨大成本，尤其是在未知开发人员、移动应用审批或设备缺乏更新功能的情况下。

API 更改通常分为非中断性或中断性。非中断性更改包括新方法和增强，而中断性更改则包括删除、修改或整体弃用。主要版本编号和迁移计划解决中断性更改，次要版本处理非中断性更改。确保更改不会中断某些应用可能较为困难，因此我们强烈建议对 API 的任何更改，即使被归类为非中断性更改，也应通过以下方式推出：

- 在发布前为新版本提供一个测试端点。
- 向开发人员发送电子邮件或其他通信，告知他们更改并提供时间和详细信息。

有效的沟通和透明的契约至关重要。分享问题详细信息、坚持承诺，并概述版本支持持续时间。例如，弃用 API 需要一种结构化方法，包括延长通知、处理媒体报道、迁移计划，以及必要时的数据导出工具。

迁移计划有利于 API 更新：

1. 引入新版本进行测试。
2. 通知用户停用旧版本。
3. 过渡期间为用户提供协助。
4. 停用旧版本。

全面的 API 管理包括预测更新和停用、有效沟通以及通过透明行动保持信任。

API 使用寿命注意事项

1. **确保对客户保证的承诺：** 这对您的 API 程序来说无疑是最关键的，您愿意向用户承诺提供什么级别的服务稳定性？为用户定义服务稳定性承诺。该承诺会影响采用。
2. **改变和打破更改过程：**确定发布程序以坚持保证。确定相关方和批准步骤。
3. **更改通信：**使用 API 定义格式检测、记录和传达更改。确保兼容性和清晰的通信。
4. **版本管理：**使用开发人员和用户 ID 监视旧版本的使用情况。制定停用流程以防止问题。
5. **产品一致性：**协调 API 的发展与相关产品的变化，考虑客户的承诺，并解决必要的调整。

维持 API 战略

这些最佳实践旨在帮助定义、实施和增强 API 战略。目标是帮助您调整战略并发现新的 API 机会。前几节中的许多问题可能需要非常详细地回答，随着时间的推移，可能会出现新的机会或风险。例如：

- 有没有办法扩大 API 对客户的价值？
- 我们是否为未来做好了充足的准备？
- 价值主张是否足以吸引开发人员真正参与？

随着 IT 环境的发展，您的 API 和产品也必须随之发展。更改的四个主要驱动因素包括：

1. **行业因素：**新的竞争对手或服务可能会取代您的 API。
2. **市场因素：**不断变化的用户需求或细分市场条件。
3. **宏观经济因素：**全球市场变化影响用户预算。
4. **趋势：**不断发展的技术或监管规定。

创建持久的 API 程序

将强大的用例与潜在客户相匹配的 API 程序具有成功启动、增长和发展的潜力。这些成功的 API 程序通过计划的 API 灵活性并通过设计和运维积极应对潜在的负面使用场景，为创新创造了空间。确保 API 条款和条件为您提供必要的工具来应对意外行为同样至关重要。

出现以下情况，您可以需要重新评估 API 战略：

- 对意外创新的依赖掩盖了价值。稳健的基础架构对于这种方法的成功至关重要。
- 缺乏对您和用户来说有说服力的用例，表明需要一种不同的方法。
- 内部对消极行为的怀疑持续存在，表明措施或沟通不足。
- 频繁的 API 损害或滥用揭示了预期价值和实际价值之间的偏差。

API 需要切合实际

不断发展的技术格局引入了一系列强大的云原生应用开发工具，包括 Kubernetes、红帽® OpenShift® 和红帽 OpenShift 服务网格，它们协同增强连接并加快出色应用的开发。然而，在这些进步中，仍要始终坚持稳健的 API 战略的基本原则和有效的 API 管理系统的重要性。当您踏上这段旅程时，请采用 API 管理解决方案，该解决方案不仅要与您的总体战略保持一致，而且还应显示出对这些平台的深刻理解或与这些平台的一致集成，以推动执行您的战略愿景。

为什么选择红帽来构建和管理 API?

红帽倡导 API 设计优先的战略方法，以确保 API 程序的成功。这种方法涵盖整个 API 生命周期，从最初的规划和设计（包括资源映射和业务场景建模）到 API 的管理（包括控制访问、访问 API 和分析使用情况）。

为了帮助开发团队使用 API 用户手册中介绍的 API 设计优先方法的最佳实践，红帽提供红帽 3scale API 管理，这是专为混合云环境构建的 API 管理解决方案。我们基于轻量级容器的分布式云原生解决方案使处理大型工作负载成为可能，并推动了以安全为中心的方法和协作。在公共云和私有云环境中共享和控制对服务、资源、应用和企业系统的访问，以便在复杂的生态系统中实现您的业务。

但是，正如我们上面提到的，API 需要切合实际并且要使用其中一些最佳实践，您需要一个完整的应用平台来进行云原生应用开发。因此，红帽 3scale API 管理包含在红帽应用基础产品组合中，该产品组合是一组中间件工具，包括集成、API 设计、API 治理和注册、流和消息传递等功能，旨在与红帽 OpenShift 容器平台配合使用。通过将红帽 OpenShift 和红帽应用基础相结合，企业拥有一个完整的云原生应用平台，可以更高效、更安全地向用户提供新软件，同时为其企业解锁战略能力和优势。

通过红帽 3scale API 管理、红帽应用基础和红帽 OpenShift 提升您的云应用开发方法，体验精简、高效和协作应用构建的未来。

结论

我们希望这些最佳实践将有助于指导您做出一些选择，或者至少能够解答您的问题。构建 API 战略的旅程始于整个企业范围内对 API 价值清晰、一致的理解。但是 API 的价值和业务目标并不是一成不变的。情况发生变化，您的 API 战略也需要随之改变。

根据我们与客户合作的经验以及对 API 经济的观察，我们总结了有价值的 API 程序的一些特点：

- 用户采用 API 是因为 API 对他们来说有价值。API 主要是为了减少困难或创造收益。
- API 继续为用户提供价值，提供根据环境变化的价值主张和战略。
- API 对企业内部来说有价值。API 完成一些重要的事情，帮助企业实现重大目标。
- 尽可能让更多利益相关者（最好是所有利益相关者）满意。

了解有关红帽 3scale API 管理的更多信息，并探索红帽的 API 管理资源。