Red Hat

API 사용 설명서

성공적인 API팀의 7가지 모범 사례

Manfred Bortenschlager

API 기반 통합 솔루션 및 API 관리를 위한
비즈니스 개발 책임자, Red Hat

Andrew Mackenzie

소프트웨어 엔지니어링 API 관리 책임자,
Red Hat

Jaya Baskaran

수석 기술 마케팅 매니저, Red Hat

Greg Pack

선임 제품 마케팅 매니저 애플리케이션
서비스 및 API 관리 솔루션, Red Hat

Red Hat

목차

핵심 요약

현대적인 애플리케이션 개발, 연결, 인프라(예: 5G 및 엣지 컴퓨팅)의 발전으로 인해 비즈니스 프로세스가 발생하는 방법과 위치가 계속 변화하고 있습니다. 이러한 발전과 함께 비즈니스 속도는 더 빨라졌지만 현대적인 애플리케이션 프로그래밍 인터페이스(API) 없이는 그 혜택을 누릴 수 없습니다.

오늘날 API는 운영 및 제품에서 파트너십 전략에 이르는 모든 요소에 새로운 기능을 추가하면서 현대적인 조직의 디지털 연결 요소로 변모했습니다. 이 e-book은 현재 API 전략을 보유하고 있거나 준비 단계에 있는 모든 조직에서 사용자와 팀에 성공적인 API 프로그램을 위한 7가지 모범 사례를 제공합니다.

소개

이 e-book을 읽기 전에 API 프로그램의 주요 목표를 떠올려 보시기 바랍니다. 나중에 나오는 질문 중 일부는 목표를 변경 또는 검증하거나 더 구체적인 내용을 제공하는 데 도움이 될 수 있습니다.

효과적인 API 프로그램은 조직의 전반적인 기업 전략을 기반으로 구축되고 목표에 기여해야 합니다. 다음 네 가지 질문에 답을 제시할 수 있다면 훌륭한 전략을 위한 기반을 갖추었다고 할 수 있습니다.

1. API를 구현하고자 하는 이유는 무엇인가?
2. 이러한 API를 사용할 것으로 예상되는 사람은 누구인가?
3. 이러한 API를 통해 실현하고자 하는 구체적인 성과는 무엇인가?
4. 성과를 실현하기 위해 어떤 방법으로 API 프로그램을 실행하고자 하는가?

API를 구현하고자 하는 이유는 무엇인가?

API를 고려할 때는 API의 가치와 용도를 포괄적으로 이해해야 합니다. 잠재적인 비즈니스 가치를 극대화하기 위해서는 일반적인 오해를 피해야 합니다. 흔한 오해는 사용자가 유료로 사용할 때만 API가 가치 있다는 것인데, API가 제품인 경우 이는 사실입니다. 그러나 대부분의 API 프로젝트는 내부적으로 유지되며 출시 속도, 재사용성, 영업, 브랜드 인지도 또는 제휴 마케팅과 같은 다양한 메트릭을 생성합니다.

전형적인 API 제공업체 활용 사례 5가지는 다음과 같습니다.

1. **옴니채널 확장.** 모바일 애플리케이션, 사물인터넷(IoT) 등과 같은 다양한 채널을 통해 접근성을 확장합니다.
2. **에코시스템 확장.** 고객 또는 파트너 에코시스템이 확장되도록 촉진합니다.
3. **도달 범위 확장.** 광범위한 거래 또는 콘텐츠 배포 네트워크를 구축합니다.
4. **신규 비즈니스 모델 강화.** 새로운 비즈니스 모델을 통해 혁신을 주도합니다.
5. **내부 혁신 생성.** 조직 내에서 혁신을 생성합니다.

대부분의 성공적인 API 전략의 예는 Amazon, Salesforce, Twilio 등을 포함한 현대적인 기술 기업에서 확인할 수 있습니다. 그러나 이러한 선두 기술 기업만이 API를 사용하여 현대화, 혁신, 파트너십 성장 또는 민첩성을 추구할 수 있는 것은 아닙니다. 대다수의 건실한 기업은 기존 시스템을 다양하게 보유하고 있으며 이러한 시스템은 하루 아침에(아마 결코) 사라지지 않습니다. API는 이러한 기존 시스템을 통합하고, 내부 또는 외부에 노출하고, 기타 서비스와 결합하여 새로운 가치를 창출하는 완벽한 접착제 역할을 합니다. 디지털 트랜스포메이션을 진행 중인 모든 비즈니스는 잘 구성된 API 전략을 구현하여 API 경제의 장점을 활용할 수 있습니다.

이러한 '이유'는 API에 대한 투자 결정을 확신하기 위한 강력한 근거를 제시할 수 있어야 합니다.

이러한 API를 사용할 것으로 예상되는 사람은 누구인가?

API 최종 사용자가 누구인지 이해하면 API 프로그램 성공 메트릭을 정의하는 데 도움이 됩니다.

- **내부 최종 사용자.** 내부 팀에서 API를 사용하여 다양한 비즈니스 목표에 필요한 회사 정보에 액세스할 수 있습니다.
- **외부 최종 사용자.** 외부의 타사 개발자가 동일한 API 또는 하위 집합을 자신의 애플리케이션에 사용할 수 있습니다.

이러한 '이유'는 API에 대한 투자 결정을 확신하기 위한 강력한 근거를 제시할 수 있어야 합니다.

API를 통해 실현하고자 하는 구체적인 성과는 무엇인가?

API로 구체적인 결과를 확인하려면 내부 및 외부의 조직적 관점을 고려해야 합니다.

- **내부 관점.** 고유하고 가치 있는 자산을 사용합니다. Meta의 "좋아요" 데이터와 같이 차별화된 데이터를 보유한 조직은 가치 있는 API를 제공할 수 있습니다.
- **외부 관점.** 시장 역학, 동향, 경쟁업체, 고객 행동을 고려합니다. 외부 요인은 비즈니스 전략을 형성하고 API 기능에 영향을 미칩니다.

시장 역학은 매핑 API에 상당한 영향을 미칩니다. 초기의 매핑 기업은 실시간 수요를 간과하여 Waze와 같은 스타트업 기업이 번창할 수 있었습니다. Google은 Waze를 인수하여 해당 기술을 성공적인 API로 통합했습니다. Twitter, Reddit 및 기타 대기업도 혁신과 에코시스템 성장을 위해 API를 채택했습니다.

올바른 API 전략은 종종 내부 자산과 외부의 시장 인사이트를 결합합니다.

성과를 실현하기 위해 어떤 방법으로 API 프로그램을 실행하고자 하는가?

마지막 질문인 '원하는 성과를 실현하기 위해 어떤 방법으로 API 프로그램을 설계할 것인가?'는 구현과 실행에 관한 질문입니다. 이를 위해서는 다음에 대한 세심한 계획이 필요합니다.

1. **기술 선택.** API를 구축하는 데 필요한 기술 스택을 결정합니다.
2. **설계 및 유지 관리.** API 설계 및 유지 관리 전략을 수립합니다.
3. **홍보 및 마케팅.** 조직 내에서 API를 홍보하고 외부에서 마케팅을 수행합니다.
4. **리소스 할당.** API 개발 및 유지 관리에 필요한 리소스를 할당합니다.
5. **팀 구성.** API 개발 및 관리를 위한 유능한 팀을 구성합니다.
6. **개발자 커뮤니티.** 외부 및 내부 개발자를 위한 전용 커뮤니케이션 계획을 세우고 유지 관리합니다.
7. **성공 메트릭.** 비즈니스 목표에 대한 성공을 추적하기 위한 방법을 수립합니다.

API와 관련하여 '왜', '누가', '무엇을', '어떻게'라는 질문에 답할 수 있는 조직은 진화하는 API 경제에서 혁신과 성장을 촉진할 수 있습니다. 이러한 질문에 대한 답은 조직마다 다르며 목표, 배포하는 전략, 그리고 매우 중요한 API 팀의 영향을 받습니다.

API 팀

API 팀은 일반적으로 다른 제품 팀과 유사한 구조로 구성됩니다. API 팀은 내부 고객이든 외부 고객이든 상관없이 다른 사용자가 사용하는 인프라를 구축, 배포, 운영, 최적화할 책임이 있습니다.

제품 팀과 마찬가지로 API 팀도 다양한 인력으로 구성될 수 있습니다. 팀에는 전략과 목표 유지를 담당할 제품 전문 담당자, API 설계 모범 사례를 보장할 설계 전문 담당자, API 기술을 코딩하는 엔지니어, 궁극적으로 API를 구동할 운영 팀 담당자가 필요합니다. 시간이 지남에 따라 지원 팀 및 커뮤니티 팀원, API 전문가, 보안 담당자 등 추가 인력이 포함될 수도 있습니다. 확장된 API 팀에는 개발자 커뮤니티의 멤버도 포함될 수 있습니다.

이 팀은 많은 인원으로 구성될 수 있지만 특히 소규모 조직에서는 일부 인원이 여러 직무를 담당하고 있을 수 있습니다. 팀원이 이해관계자의 우려 사항을 확인하는 과정을 포함하더라도 모든 이해관계자의 의견을 반영하는 것이 중요합니다.

대부분의 경우 API 팀은 임시로 구성되며 다양한 비즈니스 부문의 운영진이 포함된 다양한 조직 단위에 속할 수 있습니다. 이러한 구조로 인해 API에 대한 공유 비전을 정의하는 것이 특히 어려울 수 있습니다. 대규모 API 프로그램의 경우 다양한 API 팀이 협력해야 할 수도 있습니다.

이 e-Book에 설명된 7가지 모범 사례는 조직의 규모에 관계없이 성공적인 API 팀을 구축하는 데 도움이 되며, 예상보다 더 많은 인력이 포함될 수도 있습니다.

모범 사례 #1: 철저히 API의 가치에 중점을 둡니다

API 프로그램은 복잡성을 방지하면서 가치를 실현한다는 핵심 목표를 우선시해야 합니다. 가치 제안에 따라 중요한 API 성공 요인인 사용자 유틸리티가 결정됩니다. 강력한 가치 제안은 효과적인 마케팅을 위한 필수 요소입니다. 이러한 가치 제안을 기업의 목표에 연계하면 지속가능성을 보장하여 기존 기업이 API를 통해 제품을 개선할 수 있습니다.

Alex Osterwalder의 API 가치 모델에서는 사용자 혜택을 API 기능과 조화롭게 연계하여 사용자의 요구 사항과 API의 가치를 균형 있게 조정할 수 있는 중심을 구축합니다.[1] 요구 사항을 해결하고 고충점을 완화하며 가치를 창출하는 것이 중요합니다.

API 환경에서 의미하는 바는 무엇인가요?

이 반복 프로세스의 첫 번째 단계에서는 사용자가 수행하려는 작업을 설명합니다. 여기에는 비상 시 긴급 통신 자동 전송, 중요한 파일 백업, 특정 이벤트 감지를 위한 데이터 샘플링 등이 포함됩니다.

두 번째 단계에서는 업무를 수행하기 이전, 도중 또는 이후에 사용자에게 영향을 미치는 특정 문제를 파악해야 합니다. 여기에는 신뢰할 수 있는 다중 시도 메시지 전송 보장, 장애 감지, 다중 메시지 관리, 위치 기반 메시지 시스템 통합, 최소 대역폭을 사용하는 파일 전송 보호, 광범위한 데이터 양에 대한 실시간 상관 관계가 포함됩니다.

사용자 프로필을 구축하는 세 번째 단계에는 기타 알림 유형과 마찬가지로 위협에 대한 경고보다는 기회를 창출하는 잠재적인 장점을 개략적으로 설명하고, 충분한 신뢰성을 갖춘 경우 기타 스토리지 장비를 제거하고, 이벤트에 기반하여 조치를 자동으로 트리거하는 작업이 포함됩니다.

가치 맵을 살펴보면, 고충 해결책과 이익 창출 요인에 중점을 두고 API의 기능, 특성, 서비스를 매핑해야 합니다. 프로세스는 메시지 전달을 보장하는 메시징 API, 효율적인 버전 업데이트를 위한 스토리지 동기화 API, 구성 가능한 데이터 스트림을 제공하는 데이터 집계 API와 같은 구체적인 예시로 구성됩니다.

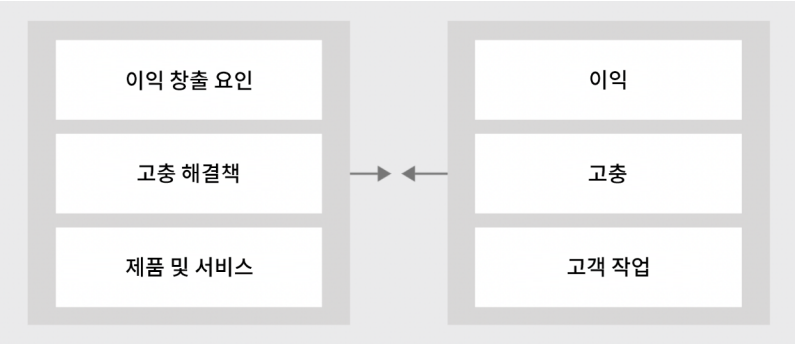

그림 1. 가치 제안 캔버스

마지막으로, API 팀은 API와 사용자 프로필 간의 적합성을 보여주는 몇 가지 설명을 작성하는 명료화 작업을 완료해야 합니다. 적합성에 대한 설명을 요약하여 하나의 포괄적인 메시지로 추상화하면 그것이 바로 API의 가치 제안이 됩니다. 이러한 '적합성' 설명은 API-사용자 연계를 더욱 공고히 하여 가치 제안을 더욱 명확하게 보여줍니다. 메시징 API를 예로 들면 이 설명은 다음과 같을 수 있습니다.

> *"고객의 의견을 여기에 인용하세요. 사이드바 콘텐츠를 추가할 때는 언어 확장에 유의하시기 바랍니다. 독일어와 같은 일부 언어는 영어보다 공간을 더 많이 차지합니다. 번역을 고려하여 사이드바의 1/4 이상을 비워 두세요."*

'단순히 내부용 API일 뿐인데 지나친 것 같다'고 생각할 수도 있습니다. 자연스러운 반응이지만 내부 활용 사례에서도 가치에 중점을 두어야 합니다. 가치 제안을 제대로 정의하지 못하면 API의 가치를 다른 팀에게 교육하고 홍보하는 데 오랜 시간이 걸릴 수 있습니다. 가치 제안을 명확히 정의하면 API 프로그램을 비즈니스의 핵심 요소로 활용할 수 있습니다.

[1] Strategyzer, '가치 제안 캔버스(Value proposition canvas).' 2023년 9월 액세스.

API 프로그램 가치에 대한 고려 사항

API 프로그램의 가치를 정의하려면 아래의 주요 영역을 고려하시기 바랍니다.

1. **사용자 식별.** 사용자 관계(고객, 파트너, 개발자), 롤, 기본 설정을 기반으로 사용자를 식별합니다.
2. **고충점 및 이익 고려.** 그림 1의 가치 제안 캔버스를 참조하면 사용자 고충, 이익, 중요한 요구 사항을 확인할 수 있습니다. 메트릭 개선 사항(속도, 수익, 비용)과 신규 기회의 잠재적 가치를 측정합니다.
3. **지원되는 활용 사례.** 가치 제안 캔버스를 사용하여 효과적인 고충 해결책과 이익 창출 요인을 식별합니다. 이러한 특정 활용 사례에 맞게 API를 설계합니다.
4. **미래 가치 확장.** 미래를 염두에 두고 가치 제안을 계획합니다. 지속적인 가치 성장을 위해 향후의 이정표, 추세 또는 기술 혁신을 예측합니다.
5. **내부 조직 가치.** API가 제공하는 내부적인 장점과 다른 팀에 대한 잠재적 가치를 평가합니다.

이러한 질문에 답하는 과정에서 API 프로그램의 명확한 가치 제안을 수립하고 사용자 요구 사항과 조직 목표에 부합하도록 할 수 있습니다.

모범 사례 #2: 처음부터 명확한 비즈니스 모델을 구축합니다

성공적인 API를 만드는 일은 가치 제안으로 충분하지 않습니다. 이를 위해서는 API와 잘 정의된 비즈니스 모델을 결합해야 합니다. API의 가치를 인식하고 전달하는 것도 중요하지만 이러한 비용은 재무 또는 실현 가능한 장점과도 균형을 이루어야 합니다. Alex Osterwalder는 공동 제작 저서인 '비즈니스 모델의 탄생(Business Model Generation)'에서 조직의 비즈니스 모델을 조직이 가치를 제안, 생성, 전달, 포착하는 방식으로 정의합니다.

API 비즈니스 모델 이해

비즈니스 모델 캔버스는 비즈니스 모델의 핵심 구성 요소를 분석한 것으로, 여기에는 다음이 포함됩니다.[2]

1. **가치 제안.** 고유한 API 가치를 정의합니다.
2. **수익원.** API가 어떻게 수익을 창출하는지 간략하게 설명합니다.
3. **비용 구조.** API 유지 관리와 관련된 비용입니다.
4. **고객 부문.** 대상 사용자 그룹입니다.
5. **고객 관계.** 조직이 사용자와 소통하는 방법입니다.
6. **채널.** 사용자에게 도달하는 배포 방법입니다.
7. **주요 파트너.** API 성공을 위한 중요 협업 파트너입니다.
8. **주요 활동.** API 운영에 필요한 핵심 태스크입니다.
9. **주요 리소스.** API 구현에 필요한 가상 자산입니다.

그림 2. 비즈니스 모델 캔버스

API는 기존 자산을 활용하여 새로운 기회를 창출할 수 있습니다. 그러나 API의 가치를 인식하는 것도 중요하지만 비용도 관리해야 합니다. 적절하지 않은 모델을 사용하면 API 가치에도 불구하고 비용이 늘어날 수 있습니다.

API 비즈니스 모델 고려 사항

비즈니스 모델을 API 사용에 연계하려면 다음을 포함한 5가지 핵심 영역을 고려해야 합니다.

1. **조직이 API를 통해 얻을 수 있는 가치.** 조직이 도달 범위를 확대하거나 혁신을 확장하는 데 API가 어떻게 도움이 되는지 살펴봄으로써 금전적 가치뿐만 아니라 다양한 가치를 평가합니다.
2. **가치 확보.** 식별된 가치를 확보하는 최적의 방법을 결정하여 가치 실현을 극대화하는 데 방해가 되는 요소를 최소화합니다.
3. **비용 충당.** 엔지니어링 또는 마케팅과 같이 API 팀 이외에서 종종 발생하는 비용을 포함하여 API 관련 비용을 파악하고 이러한 비용을 위한 계획을 세웁니다.
4. **장기적인 노력.** API는 초기 투자 이외에도 운영 및 유지 관리를 위해 지속적으로 노력해야 한다는 점을 이해합니다.
5. **전략적 파트너십.** 중요한 파트너십을 확인하고 API 개발 및 시장 진입 시 파트너와 공급업체가 제공하는 보완 제품을 활용합니다.

모범 사례 #3: 설계 및 구현 시 사용자 고려

API 설계에서는 일관된 사용자 환경을 위해 기본적인 설계 원칙을 적용합니다. 숙련된 개발자가 API를 직관적으로 이해할 수 있는 '가동 준비 완료' 상태를 확보하는 것이 설계의 목표입니다. API 설계는 두 가지 핵심 영역, 즉 간소화와 유연성에 집중해야 합니다.

간소화

API 설계에서 간소화는 상황에 따라 다르며 활용 사례에 따라 설계의 복잡성이 달라집니다. API 메서드 세분화에서 올바른 균형을 유지하는 것이 중요합니다. 간소화는 다양한 수준에서 고려할 수 있습니다.

1. **데이터 형식.** XML, JSON, 독점적 형식 또는 이들의 조합 중에서 선택합니다.
2. **메서드 구조.** 메서드는 일반적인 것부터 매우 구체적인 것까지 다양하며 일반적으로 특정 활용 사례를 달성하도록 순서가 지정됩니다.
3. **데이터 모델.** 노출된 API 데이터 모델은 기본 모델과 다를 수 있으며 유용성과 유지 관리 용이성에 영향을 미칠 수 있습니다.
4. **인증.** 인증 메커니즘에 따라 각각의 강점과 약점이 있으며 상황에 따라 선택할 수 있습니다.
5. **사용 정책.** 개발자 권한과 할당량을 이해하고 관리할 수 있어야 합니다.

유연성

간소화와 유연성을 균형 있게 유지하는 것이 매우 중요합니다. 지나치게 단순한 API는 특정 활용 사례에만 적합할 수 있어 적응성이 제한될 수 있습니다. 유연성을 확립하려면 기본 시스템 및 데이터 모델을 포함하여 잠재적 운영 공간을 간략히 설명합니다. 이러한 운영의 실행 가능하고 가치 있는 하위 집합을 정의합니다. 간소화와 유연성 간의 균형을 적절하게 조정하려면 다음을 고려합니다.

1. **최소 단위 운영 노출.** 최소 단위 운영을 결합하여 전체 운영 공간을 다룹니다.
2. **일반적이고 가치 있는 활용 사례 식별.** 최소 단위 운영을 결합하는 메타 운영의 두 번째 레이어를 설계하여 이러한 활용 사례를 지원합니다.

API 프로토콜 설계 고려 사항

REST(Representational State Transfer) 아키텍처 스타일은 여전히 API 개발을 위한 주요 표준이지만 API 프로토콜은 훨씬 더 다양해지고 있습니다. 최근에는 스트리밍 API 또는 WebSocket과 같은 개념이 등장했습니다. 클래식 웹 서비스 API도 계속 유지될 것입니다. 프로토콜을 선택할 때는 사용자에게 가장 관련성이 높은 프로토콜을 선택한 다음 간소화와 유연성이 적절하게 균형을 이루도록 한다는 원칙을 따라야 합니다. REST API를 제공하는 인프라는 계속 발전하고 있습니다.

API 설계를 더 세부적으로 살펴보려면 다음 영역을 고려합니다.

1. **활용 사례와 연계.** 확인된 활용 사례를 지원하면서도 혁신적이고 발생 빈도가 낮은 시나리오에 대한 유연성을 유지하도록 API를 설계합니다.
2. **신중하게 RESTful API 고려.** RESTful API는 최신 방식이지만 특정 활용 사례에서는 다른 아키텍처 스타일이 더 적합할 수 있으므로 실제 요구 사항에 적합한지를 평가해야 합니다.
3. **추상화된 데이터 모델.** API와 데이터 모델 사이에 추상화 계층이 있는 API를 구현하여 필요한 경우 외에는 데이터베이스에 직접 액세스하지 않도록 합니다.
4. **지리적 고려 사항.** 주요 사용자 지역 근처에 데이터센터를 전략적으로 배치하여 대기 시간 및 가용성과 같은 비기능적 측면을 고려합니다.
5. **제품과의 통합.** 구조의 내부 및 외부 통신을 명확하게 유지하면서 조정 또는 분리를 통해 API 설계가 다른 제품과 조화를 이루도록 합니다.

API 구현 및 배포 인프라

조직이 데이터 및 애플리케이션을 위해 클라우드 및 하이브리드 클라우드 인프라로 전환함에 따라 API 구현 및 배포 인프라도 더욱 다양해졌습니다. API는 마이크로서비스를 기반으로 구축할 수 있으며 모놀리스와 그 사이의 모든 방식으로 구축된 API와 결합될 수 있습니다. 컨테이너, 가상 머신(VM), 베어 메탈 또는 퍼블릭 클라우드 환경 어딘가에 존재할 수 있습니다.

기타 애플리케이션과 마찬가지로 API 라이프사이클 관리에 있어 자동화된 지속적 통합/지속적인 제공(CI/CD) 파이프라인을 보유하는 것이 매우 중요합니다. GitOps 및 Argo CD를 채택하면 중앙집중식 API 구성 관리와 자동화된 배포 CD를 제공하고 API 팀 전체의 협업 노력을 강화하여 개발 속도를 높이고 API 제품 품질을 향상할 수 있습니다.

모범 사례 #4: API 운영을 우선시합니다

API 플랫폼 팀은 API가 실행된 후 이에 액세스 가능하고 개발자의 기대에 따라 제공되도록 관리합니다. 벤더가 솔루션을 제공하지만 성공을 위해서는 올바른 전략을 선택하는 것이 중요합니다. API 관리에는 2가지 주요 기능이 포함됩니다.

1. 내부 프로세스를 간소화하여 효율성을 높이고 비용을 절감합니다.
2. 효과적인 운영을 통해 프로그램에 대한 외부 개발자의 기대를 충족합니다.

뛰어난 API 구축 1부: 표준(Building Great APIs Part 1: The Gold Standard)과 그림 3의 API 운영 도넛 차트는 이러한 목표를 달성하는 데 도움이 될 수 있습니다.

API 운영 도넛 차트

API 운영 도넛 차트는 조직의 API 전략을 달성하는 데 필요한 운영 전술을 정의하는 데 사용할 수 있습니다. 도넛 차트의 안쪽 원은 조직의 내부 활동과 효과를 나타내고, 링 외부의 모든 요소는 외부 효과를 나타냅니다.

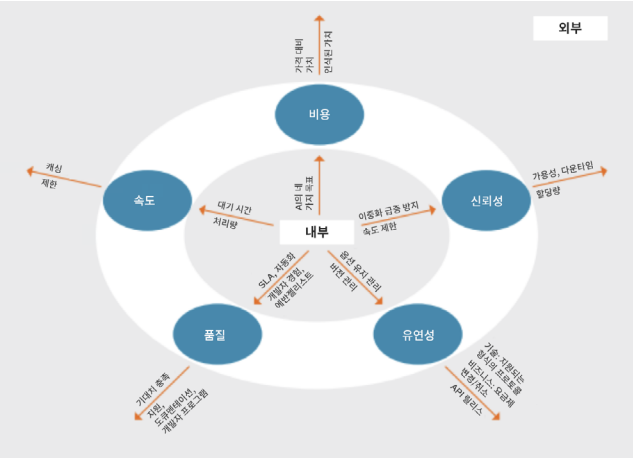

그림 3. API 운영 도넛 차트

1. **신뢰성.** 이중화 또는 할당량 및 비율 제한을 통해 API 가용성을 보장합니다. 이들은 비즈니스 모델에 부합하며 다운타임을 예방합니다.
2. **유연성.** 요금제 간 변경이나 취소 등 기술 및 비즈니스 채택 옵션을 제공합니다. 유연성이 향상되면 내부의 작업과 비용이 늘어날 수 있다는 점에 유의해야 합니다.
3. **품질.** 서비스 수준 계약(SLA)과 간소화된 프로세스를 사용하여 개발자의 기대치를 일관되게 충족합니다.
4. **속도.** 비즈니스 모델에 잠재적으로 부합하도록 제한 및 캐싱과 같은 기술을 사용하여 대기 시간을 단축하고 처리량을 늘립니다.
5. **비용.** 품질 저하 없이 내부 비용을 최소화하면서 개발자 가치를 최적화합니다.

이처럼 다양한 운영 과제를 지원하기 위한 기술 인프라를 제공하는 여러 벤더가 있으며 Red Hat도 그중 하나입니다. 일반적으로 벤더를 이용하는 것이 이러한 문제를 해결하는 비용 효과적인 방법이지만 이러한 전략은 철저해야 합니다.

필수 API 관리 구성 요소

API 에코시스템을 운영하려면 효과적인 관리를 위해 고유의 구성 요소 집합이 필요합니다. 이러한 구성 요소는 API 전략에 따라 다르지만 3가지 핵심 구성 요소는 동일합니다.

1. **액세스 제어.** 액세스를 허용하고 들어오는 트래픽을 식별하는 인증 및 권한 부여 시스템을 구현합니다.
2. **요금 제한 및 사용 정책.** 트래픽에 할당량과 제한을 적용하여 부하를 예측합니다.
3. **분석.** API 사용 모니터링을 위해 트래픽 패턴을 포착하고 분석합니다.

API 운영 고려 사항

API 운영 전략이 조직의 전반적인 전략과 조화를 이루도록 하면 API의 중요성에 따라 리소스를 할당하는 데 도움이 됩니다.

API 운영 계획을 더 세부적으로 살펴보려면 다음 영역을 고려합니다.

1. **액세스 제어.** 누가 액세스하고, 작업을 수행하고, 제한을 적용할지를 정의하여 보안 중심의 API 사용을 보장합니다.
2. **메트릭 및 경고.** 분석, 맞춤형 메트릭 측정, API 성능에 대한 경고 설정을 통해 가시성을 확보합니다.
3. **급증 관리.** 액세스 제어 및 다양한 정책을 사용하여 인프라를 계획하고 급증 방지 또는 제한과 같은 대체 메커니즘을 설정합니다.
4. **API 가동 시간 책임.** API 가동 시간에 대한 소유권을 명확히 하는 것은 가치 창출 및 확보와 직접적으로 연결되므로 매우 중요합니다.
5. **원치 않는 사용 처리.** 예상되는 사용과 예상치 못한 사용을 구분하고 사전 예방적인 운영과 이용 약관을 통해 관리합니다.
6. **커뮤니케이션.** 내부 및 외부 개발자에게 계획된 다운타임, 유지 관리, API 변경 사항을 알릴 수 있도록 명확한 커뮤니케이션 계획을 마련합니다.

모범 사례 #5: 개발자 경험에 집중합니다

개발자 경험은 API 설계에 관한 것처럼 들릴 수 있지만 실제로는 그 이상을 의미합니다. 개발자 경험을 API 자체가 아닌 API 패키징과 제공 측면에서 생각해 보세요. 훌륭하게 설계된 REST 또는 컨테이너화된 API를 보유하더라도 액세스를 위한 등록, 문서 읽기, 테스트를 수행하기 어렵다면 개발자 환경이 제대로 구성되지 않았음을 나타내며 이는 API 성공에 큰 영향을 미칠 수 있습니다.

성공적인 API는 훌륭한 설계 그 이상입니다

간소화와 유연성을 갖추어 API를 설계했다 하더라도 개발자가 API를 활용하지 않고 결과적으로 채택하지 않으면 무용지물이 됩니다. 반면 세심하게 고려한 API 설계는 개발자 경험과 채택에 상당한 영향을 미칩니다. 채택은 개발자 경험의 필수적인 부분입니다.

API 및 API 관리 분야의 선구자 중 한 명인 John Musser는 다음과 같이 참여를 높이는 데 여전히 유효한 몇 가지 방법을 설명했습니다.

1. API의 용도를 명확하게 정의합니다.
2. 간편한 등록과 무료 액세스를 제공합니다.
3. 가격을 투명하게 전달합니다.
4. 철저한 도큐멘테이션을 제공합니다.

용이하게 채택할 수 있도록 API 설계를 개선하는 데 있어 핵심 메트릭은 TTFHW(Time To First Hello World)로, 이는 초기 API 상호 작용을 측정합니다. 더 광범위한 맥락을 설명하는 것으로는 개발자 프로그램을 강조하는 TTFPA(Time To First Profitable App)가 있습니다. 수익성은 API 및 비즈니스 전략에 따라 정의되는 문제이기 때문에 이는 더 까다로울 수 있습니다. API 프로그램의 일부로 API 운영을 고려해야 하기 때문에 TTFPA를 고려하는 것이 도움이 됩니다.

개발자 경험에는 가치 있는 제품 또는 서비스 설계와 손쉬운 접근성이라는 두 가지 원칙이 포함됩니다. 포털, 커뮤니티, 에반젤리스트, 이벤트, 측정을 포함한 견고한 개발자 참여 프로그램을 통해 참여를 높일 수 있습니다. 개발자 포털에 대한 일반적인 메트릭의 예로는 페이지 방문 수, 등록 수, API 트래픽 또는 지원 요청이 있습니다. 이벤트는 참석자 수, 해커톤에서의 API 채택 또는 리드로 측정할 수 있습니다. "이벤트에서의 대화가 API 등록 증가로 이어졌나요?"와 같은 상관 관계를 만드는 것이 도움이 됩니다.

개발자 프로그램에는 다음 요소가 포함되어야 합니다.

- **커뮤니티 구축.** 해커톤과 같은 이벤트를 통해 상호 작용을 촉진하고 채택을 홍보합니다.
- **개발자 에반젤리스트.** 성공에 반드시 필요하며 API의 장점을 활용하고 홍보합니다.
- **파일럿 파트너 및 고객 사례.** 얼리 어답터가 피드백과 고객 사례를 제공합니다.
- **에코시스템 파트너.** 파트너십은 시너지 효과를 통해 채택을 늘립니다.
- **측정.** API 목표에 연계된 메트릭을 통해 효과적으로 관리할 수 있도록 지원합니다.
- **통신.** 포털, 뉴스레터, 프라이빗 Slack 또는 Discord 서버를 통해 이루어집니다.

대상에 맞춤화된 개발자 프로그램을 제작하면 참여와 경험이 강화됩니다.

개발자 경험 평가를 위한 고려 사항

세심하게 고려한 개발자 경험을 제공하면 개발자가 자신의 잠재력을 극대화하도록 영감을 줄 수 있습니다. 다음은 개발자 경험을 평가하는 데 필수적인 6가지 고려 사항으로, 단순한 코드 라인이 아닌 창의성과 생산성을 높여주는 관문 역할을 하는 API를 구축하는 데 도움이 됩니다.

1. **가치 설명.** 개발자에게 API의 가치를 전달하기 위해 짧은 엘리베이터 피치를 생성합니다.
2. **TTFHW 및 TTFPA.** 모든 개발자 경험 요소(예: 포털)를 고려하여 TTFHW와 TTFPA를 평가하고 이를 최소화합니다.
3. **온보딩 프로세스.** 온보딩을 API 활용 사례에 맞게 조정하여 초기 개발자 성공을 위해 간소화합니다.
4. **가치 제공.** API가 충분한 가치를 제공하여 개발자를 유치하고 유지할 수 있도록 합니다.
5. **개발자 지원.** 더 심각한 문제에 대한 추가 메커니즘과 함께 도큐멘테이션, FAQ, 포럼을 통해 셀프 서비스 지원의 우선순위를 정합니다.
6. **독창적인 사용 방식.** 비표준 활용 사례를 탐색하는 개발자를 위한 지원 및 도큐멘테이션을 다룹니다.

모범 사례 #6: 마케팅 101 그 이상을 수행합니다

개발자를 대상으로 API를 마케팅하는 일은 어려울 수 있습니다. 특히 제시된 가치가 개발자의 비즈니스 또는 기술 요구 사항과 일치하지 않는 경우 더욱 그렇습니다. API는 다른 제품과 마찬가지로 세분화, 타겟팅, 포지셔닝(STP)에 맞춰 마케팅해야 합니다. 효과적인 마케팅은 API 가치와 STP 원칙에 따라 올바른 API와 올바른 개발자를 연결합니다.

1. **세분화(Segmentation).** 고객을 내부 사용자, 파트너, 최종 사용자 또는 외부 개발자로 분류합니다. 효과적인 참여를 위해 JTBD(Jobs-to-be-done) 방법을 사용하여 광범위한 개발자 시장을 세분화합니다.
2. **타겟팅(Targeting).** 접근성, 실질성, 차별화를 기반으로 부문별 소구점을 평가합니다. 유망한 부문을 선택하고 해당 부문에 따라 마케팅 전략을 조정합니다.
3. **포지셔닝(Positioning).** 특정 요구 사항을 해결하고, 문제점을 완화하고, 선택한 개발자 부문에 장점을 제공하여 API를 돋보이게 만듭니다.

마케팅에서 개발자 경험의 우선순위를 지정하는 것은 성공에 매우 중요합니다. 개발자 에반젤리스트, 강력한 개발자 포털, 해커톤 및 기타 이벤트와 같은 전략을 통해 개발자와 견고한 관계를 구축할 수 있습니다.

API 마케팅을 위한 고려 사항

1. **대상.** 내부 사용자, 파트너, 고객 또는 일반 대중에 따라 API를 단계별로 발전시키며 핵심 사용자 그룹의 우선순위를 지정합니다.
2. **전문가 선정.** 효과적인 홍보를 위해 엔지니어링, 지원, 영업, 제품 관리 등의 영역을 고려하여 API의 가치 제안에 맞는 전문가를 선택합니다.
3. **이벤트 전략.** API 목표에 따라 이벤트 유형(수평 또는 수직, 글로벌 또는 로컬, 컨퍼런스 또는 해커톤)을 선택합니다.
4. **해커톤 적합성.** 해커톤 관련성(등록, SDK 다운로드, 애플리케이션, 채용, 브랜딩)을 평가하고 이에 따라 계획을 세웁니다.
5. **내부 마케팅.** 여러 부서에 걸쳐 지원을 확보하고 마케팅 부서와 이를 명확히 하여 제품 팀과 고객에게 장점을 전달합니다.

모범 사례 #7: API 폐기 및 변경 관리를 고려합니다

API 조언은 API 설계, 생성, 운영에 중점을 두는 경향이 있습니다. 그러나 API 여정에서 간과되는 가장 중요한 부분 중 하나는 출시 및 운영 수개월 후에 발생하는데, 바로 API 사용 중단을 포함한 API 업데이트 관리입니다.

갑작스러운 변경으로 인한 중단은 신뢰를 떨어뜨리고 상당한 비용을 야기할 수 있습니다. 특히 알 수 없는 개발자, 모바일 애플리케이션 승인 또는 업데이트 기능이 없는 기기의 경우 더욱 그러합니다.

API 변경은 일반적으로 호환 준수(nonbreaking) 변경 또는 단절적(breaking) 변경으로 분류됩니다. 호환 준수 변경에는 새로운 메서드와 향상된 기능이 포함되는 반면, 단절적 변경에는 제거, 수정 또는 전체 사용 중단이 포함됩니다. 메이저 버전 번호 및 마이그레이션 계획은 단절적 변경을 다루고, 마이너 버전은 호환 준수 변경을 다룹니다. 변경으로 인해 일부 애플리케이션이 단절되지 않도록 하는 것이 어려울 수 있으므로, API에 대한 변경은 호환 준수 변경으로 분류되더라도 다음과 같이 롤아웃해야 합니다.

- 출시 전에 새 버전을 사용하여 테스트 엔드포인트를 제공합니다.
- 개발자에게 이메일이나 기타 커뮤니케이션을 보내 변경을 알리고 시기와 세부 정보를 제공합니다.

효과적인 커뮤니케이션과 투명한 계약이 중요합니다. 문제에 대한 세부 정보를 공유하고 약속을 지키며 버전 지원 기간을 간략히 설명합니다. 예를 들어, API를 사용 중단하려면 사전 공지 확장, 언론 보도 다루기, 마이그레이션 계획, 필요 시 데이터 내보내기 툴 등을 포함한 체계적인 접근 방식이 필요합니다.

마이그레이션 계획은 다음과 같이 API 업데이트를 용이하게 합니다.

1. 테스트를 위해 새 버전을 도입합니다.
2. 이전 버전 사용 종료에 대해 사용자에게 알립니다.
3. 전환하는 동안 사용자를 지원합니다.
4. 이전 버전을 사용 종료합니다.

포괄적인 API 관리에는 업데이트 및 기간내 사용(retirement) 예측, 효과적인 커뮤니케이션, 투명한 조치를 통한 신뢰 유지가 포함됩니다.

API 수명 고려 사항

1. **고객 보증에 대한 약속 보장.** 이는 API 프로그램에 있어 가장 중요한 사항입니다. 사용자에게 어느 수준의 서비스 안정성을 제공하시겠습니까? 사용자를 위한 서비스 안정성 약속을 정의합니다. 이 약속은 채택에 영향을 미칩니다.
2. **변경 및 단절적 변경 프로세스.** 보증을 유지하기 위한 릴리스 절차를 결정합니다. 관련 당사자 및 승인 단계를 식별합니다.
3. **변경 관련 커뮤니케이션.** API 정의 형식을 사용하여 변경 사항을 감지, 문서화, 전달합니다. 호환성과 명확한 커뮤니케이션을 보장합니다.
4. **버전 관리.** 개발자 및 사용자 ID를 사용하여 이전 버전의 사용을 모니터링합니다. 기간내 사용(retirement) 프로세스를 구축하여 문제를 방지합니다.
5. **제품 조정.** 관련 제품 변경 사항에 따라 API 발전을 조정하고, 고객과의 약속을 고려하고, 필요한 조정을 처리합니다.

API 전략 지속

이러한 모범 사례는 API 전략을 정의, 구현, 향상하는 데 도움을 주기 위한 것입니다. 목표는 전략을 조정하고 새로운 API 기회를 발견하도록 돕기 위한 것입니다. 이전 섹션에 있는 질문 대부분은 자세한 답변이 필요할 수 있으며, 시간이 지남에 따라 새로운 기회나 위험이 발생할 수 있습니다. 예를 들면 다음과 같습니다.

- API의 가치를 고객에게 확장할 수 있는 방법이 있습니까?
- 미래를 위해 충분히 준비가 되어 있습니까?
- 개발자의 실제 참여를 이끌어 낼 수 있을 만큼 매력적인 가치 제안입니까?

IT 환경이 발전함에 따라 API와 제품도 발전해야 합니다. 변화의 네 가지 주요 요인은 다음과 같습니다.

1. **산업 요인.** 새로운 경쟁업체나 서비스가 귀사의 API를 대체할 수 있습니다.
2. **시장 요인.** 사용자 수요 또는 시장 부문의 조건이 변화합니다.
3. **거시경제적 요인.** 글로벌 시장 변화는 사용자 예산에 영향을 미칩니다.
4. **추세.** 기술 또는 규제 의무 사항이 진화하고 있습니다.

지속되는 API 프로그램 개발

잠재 고객과 강력한 활용 사례를 연결하는 API 프로그램에는 성공적으로 출시, 성장, 발전할 수 있는 잠재력이 있습니다. 이처럼 성공적인 API 프로그램은 계획된 API 유연성을 통해 혁신을 위한 가능성을 확보하고 설계 및 운영을 통해 발생할 수 있는 부정적인 사용 시나리오를 사전에 해결합니다. API 이용 약관에서 예상치 못한 동작에 대응하는 데 필요한 툴을 제공하는지 확인하는 것도 중요합니다.

다음과 같은 경우 API 전략을 재평가해야 할 수 있습니다.

- 예상치 못한 혁신에 의존하면 가치를 무색하게 만듭니다. 이러한 접근 방식으로 성공하기 위해선 강력한 인프라가 중요합니다.
- 귀사와 사용자를 위한 강력한 활용 사례가 부족하다는 것은 다른 접근 방식이 필요함을 의미합니다.
- 부정적인 동작에 대해 내부적으로 의심이 지속된다면 충분한 조치나 커뮤니케이션이 이루어지지 않았음을 나타냅니다.
- 빈번한 API 손상이나 오용은 의도한 가치와 실제 가치가 일치하지 않음을 나타냅니다.

외부와 단절된 환경에서는 API가 존재하지 않습니다

끊임없이 진화하는 기술 환경은 쿠버네티스, Red Hat® OpenShift®, Red Hat OpenShift Service Mesh를 비롯하여 다양하고 강력한 클라우드 네이티브 애플리케이션 개발 툴을 도입했습니다. 이러한 툴은 시너지 효과를 발휘하여 연결성을 강화하고 탁월한 애플리케이션의 개발을 가속화합니다. 그러나 이러한 발전 속에서도 강력한 API 전략의 기본 원칙과 효과적인 API 관리 시스템의 중요성은 변하지 않습니다. 이 여정을 시작할 때 전반적인 전략과 일치할 뿐만 아니라 전략적 비전 실행을 촉진하기 위해 이러한 플랫폼에 대한 강력한 이해 또는 일관된 통합을 보여주는 API 관리 솔루션을 찾아야 합니다.

API를 구축하고 관리하는 데 Red Hat을 선택해야 하는 이유

Red Hat은 API 프로그램의 성공을 보장하기 위해 전략적 API 설계 우선 접근 방식을 지지합니다. 이 접근 방식에서는 리소스 매핑 및 비즈니스 시나리오 모델링을 포함하는 초기 계획 및 설계부터 액세스 제어, API 액세스, 사용 분석을 포함하는 API 관리에 이르기까지 전체 API 라이프사이클을 다룹니다.

Red Hat은 본 API 사용 설명서에서 제시한 API 설계 우선 접근 방식에 대한 이러한 모범 사례를 통해 개발 팀을 지원할 수 있도록 하이브리드 클라우드 환경을 위해 특별히 구축된 API 관리 솔루션인 Red Hat 3scale API Management를 제공합니다. 분산형 경량 컨테이너 기반 클라우드 네이티브 솔루션을 사용하면 대규모 워크로드를 처리하고 보안 중심 접근 방식과 협업을 촉진할 수 있습니다. 퍼블릭 및 프라이빗 클라우드 환경 전반에 걸쳐 서비스, 리소스, 애플리케이션, 엔터프라이즈 시스템에 대한 액세스를 공유하고 제어하면 복잡한 에코시스템에서도 비즈니스를 강화할 수 있습니다.

그러나 위에서 언급한 것처럼 API는 외부와 단절된 환경에서 존재하지 않으며 이러한 모범 사례 중 일부를 사용하려면 클라우드 네이티브 애플리케이션 개발을 위한 완전한 애플리케이션 플랫폼이 필요합니다. 따라서 Red Hat 3scale API Management는 Red Hat OpenShift Container Platform에서 작동하도록 설계된 통합, API 설계, API 거버넌스 및 레지스트리, 스트리밍, 메시징과 같은 기능이 포함된 미들웨어 툴 모음인 Red Hat Application Foundations 포트폴리오에 포함되어 있습니다. Red Hat OpenShift와 Red Hat Application Foundations를 함께 사용하는 조직은 새로운 소프트웨어를 사용자에게 더욱 효율적이고 안전하게 제공하는 동시에 조직을 위한 전략적 기능과 장점을 활용할 수 있는 완벽한 클라우드 네이티브 애플리케이션 플랫폼을 갖추게 됩니다.

Red Hat 3scale API Management, Red Hat Application Foundations, Red Hat OpenShift를 통해 클라우드 네이티브 애플리케이션 개발 접근 방식을 개선하여 간소화되고 효율적이며 협업이 가능한 애플리케이션 구축의 미래를 경험해 보세요.

결론

소개해 드린 모범 사례가 선택이나 궁금한 사항에 도움이 되기를 바랍니다. API 전략을 구축하는 여정은 API 가치에 대한 명확하고 일관된 전사적인 이해를 확보하는 것에서 시작됩니다. 그러나 API의 가치와 비즈니스 목표는 고정되어 있지 않습니다. 상황은 변하고 이에 따라 API 전략도 변해야 합니다.

고객과 협력하고 API 경제를 관찰한 경험을 바탕으로 가치 있는 API 프로그램의 몇 가지 특성을 다음과 같이 요약할 수 있습니다.

- 사용자는 가치가 있기 때문에 API를 채택합니다. 이는 고충을 완화하거나 이익을 창출함으로써 필수적인 작업을 수행합니다.
- API는 환경에 따라 변화하는 가치 제안과 전략을 제공하며 사용자에게 지속적으로 가치를 제공합니다.
- API는 내부적으로 조직에 가치를 제공합니다. 중요한 역할을 수행하고 조직이 중요한 목표를 달성하도록 돕습니다.
- 가능한 한 많은(이상적으로는 모든) 이해관계자가 만족합니다.

Red Hat 3scale for API management에 대해 자세히 알아보고 Red Hat의 API 관리 리소스를 살펴보시기 바랍니다.