

Apigee API Management

Sesión 02

Ricardo Montecino

ricardo.montecino@sansano.usm.cl

Agenda

- Introducción
- Apigee Fundamentals
- API Design (best practices)
- Apigee Edge
- Laboratorio practico OpenAPI Spec

Introducción

Presentación y road map del curso

Instructor

- Ricardo Montecino Olivares.
- Ingeniero en Informática DUOC UC.
- Magíster en Tecnologías de la Información Universidad Técnica Federico Santa María.
- Founder Facturaya <https://facturaya.cl>
- Platform Architect en Caja Los Andes.
- Más de 10 años de experiencia en Tecnologías de la Información.
- LinkedIn: <https://linkedin.com/in/rmontecinonet>
- Email: ricardo.montecino@sansano.usm.cl

Roadmap

1. API Management.
 - 1.1 Introduction to APIs.
 - 1.2 API Platforms.
 - 1.3 API Management Fundamentals.
2. Apigee Design.
 - 2.1 Apigee Fundamentals
 - 2.2 API Design
 - 2.3 Best Practices.
 - 2.4 Open API.
 - 2.5 Practice Lab.
3. Apigee Development.
 - 3.1 API Proxy.
 - 3.2 API Policies.
 - 3.3 Target Servers.
 - 3.4 API Products.
 - 3.5 Debugging APIs.
 - 3.6 Practice Lab.
4. Security.
 - 4.1 API Keys.
 - 4.2 OAuth 2.0.
 - 4.3 TLS.
 - 4.4 Practice Lab.
5. Monitoring.
 - 5.1 API Metrics.
 - 5.2 Analytics.
 - 5.3 Alerts.
6. Additional Practice Lab (optional).

API Management

Apigee Design

Apigee Development

Security

Monitoring

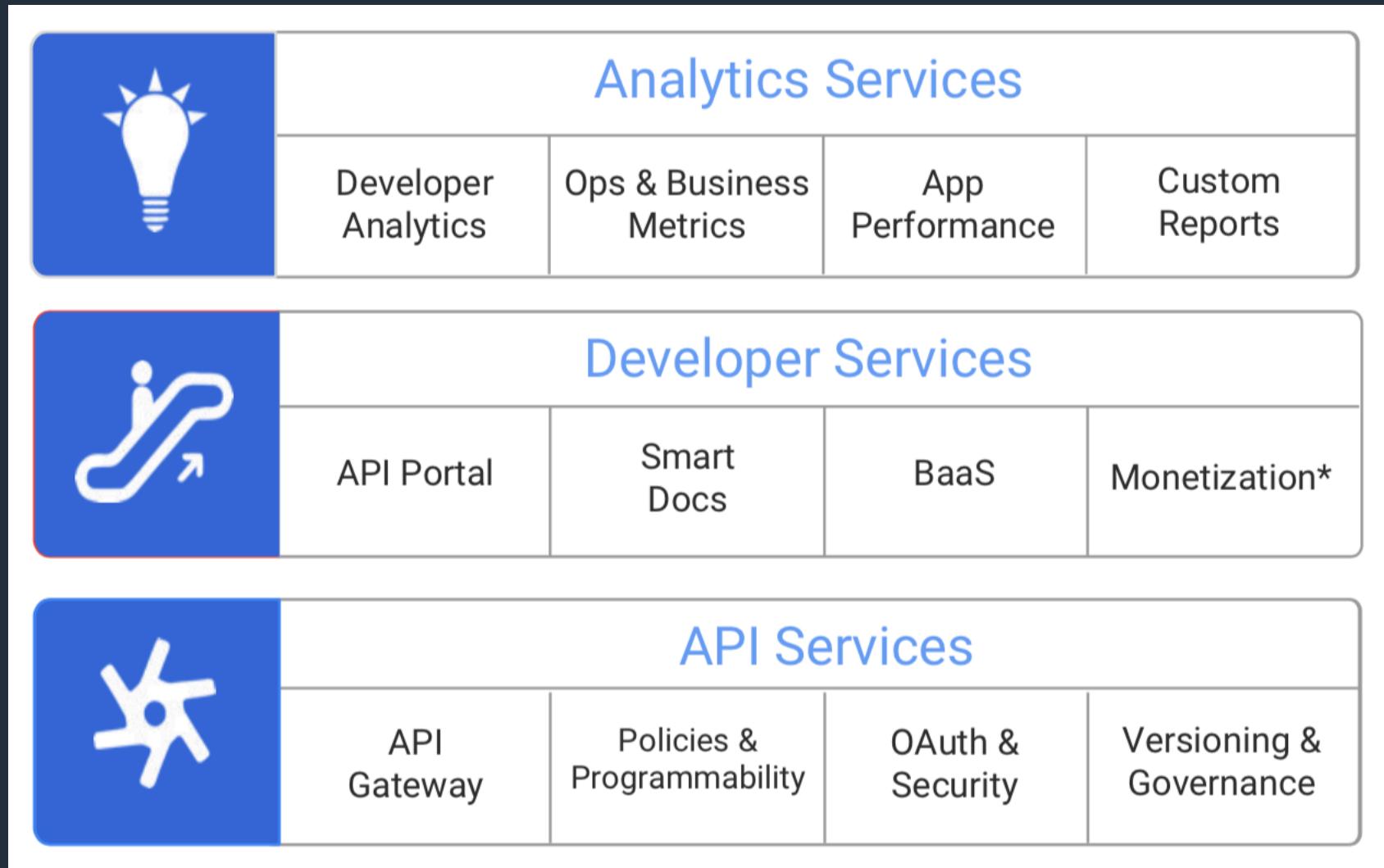
Objetivos específicos sesión

1. Conocer los fundamentos de Apigee como API Management. Sus distintos productos tales como: Apigee Edge, Apigee Sense y Apigee Monetization.
2. Comprender la anatomía de Apigee Edge. Entendiendo sus conceptos y flujos principales.
3. Interiorizarse respecto de mejores prácticas de diseño APIs, utilizando Apigee Edge.
4. Revisar API guidelines de mercado, en el contexto de mejores prácticas de diseño APIs.
5. Practicar con la plataforma Apigee Edge, para diseñar con estándar OpenAPI.



Apigee Fundamentals

Apigee Edge



Apigee Sense



Intelligent API Security



Behavior Detection



API Protection

Apigee Monetization



Various revenue models



Payment & billing capabilities

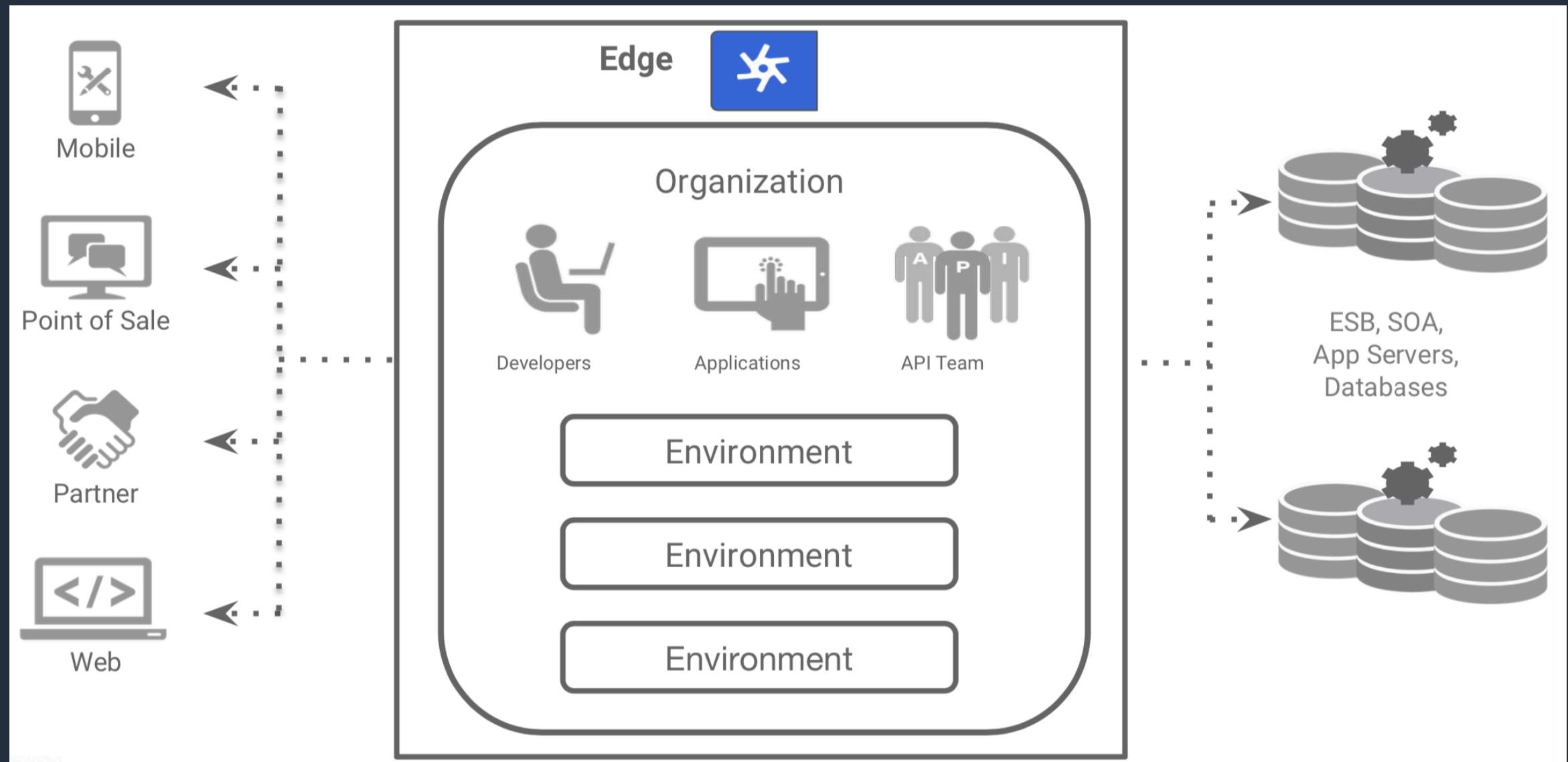


Flexible reporting

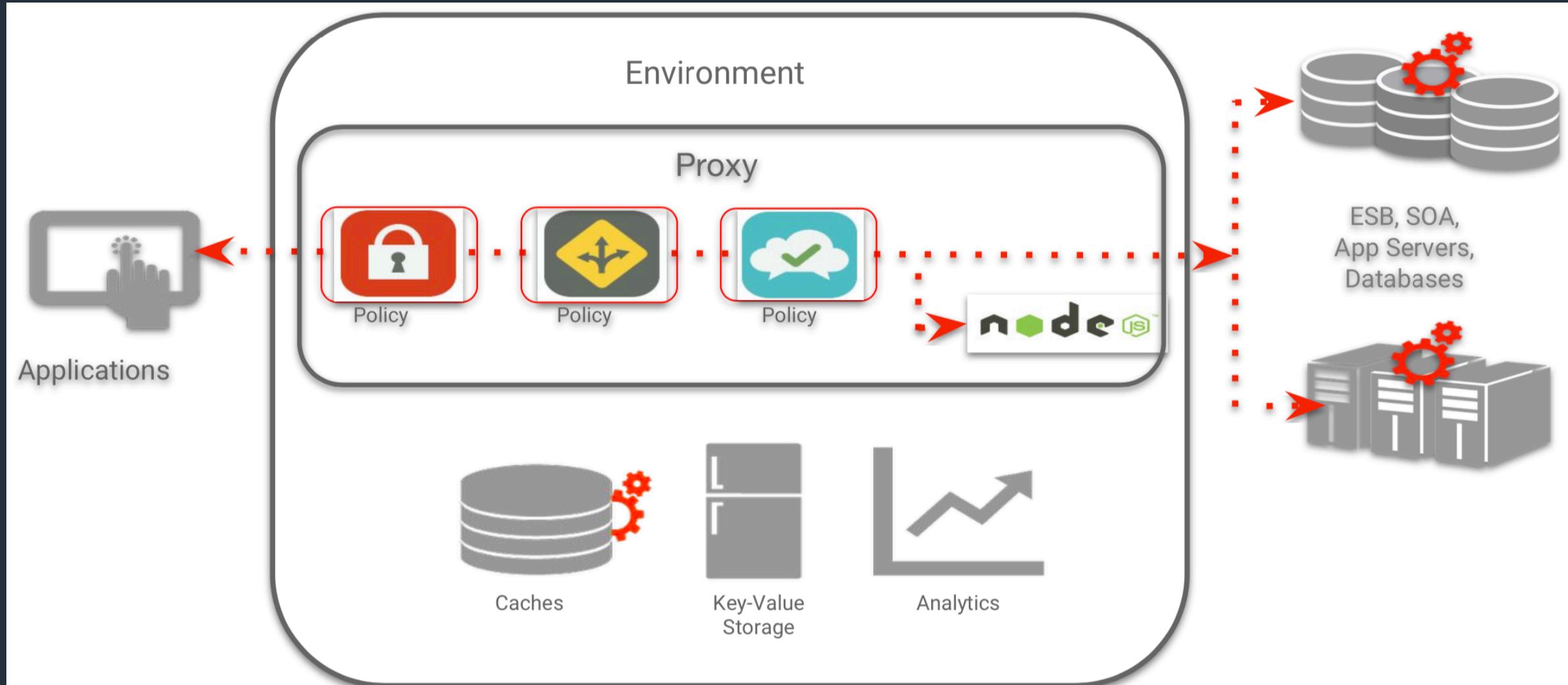


Developer Portal integration

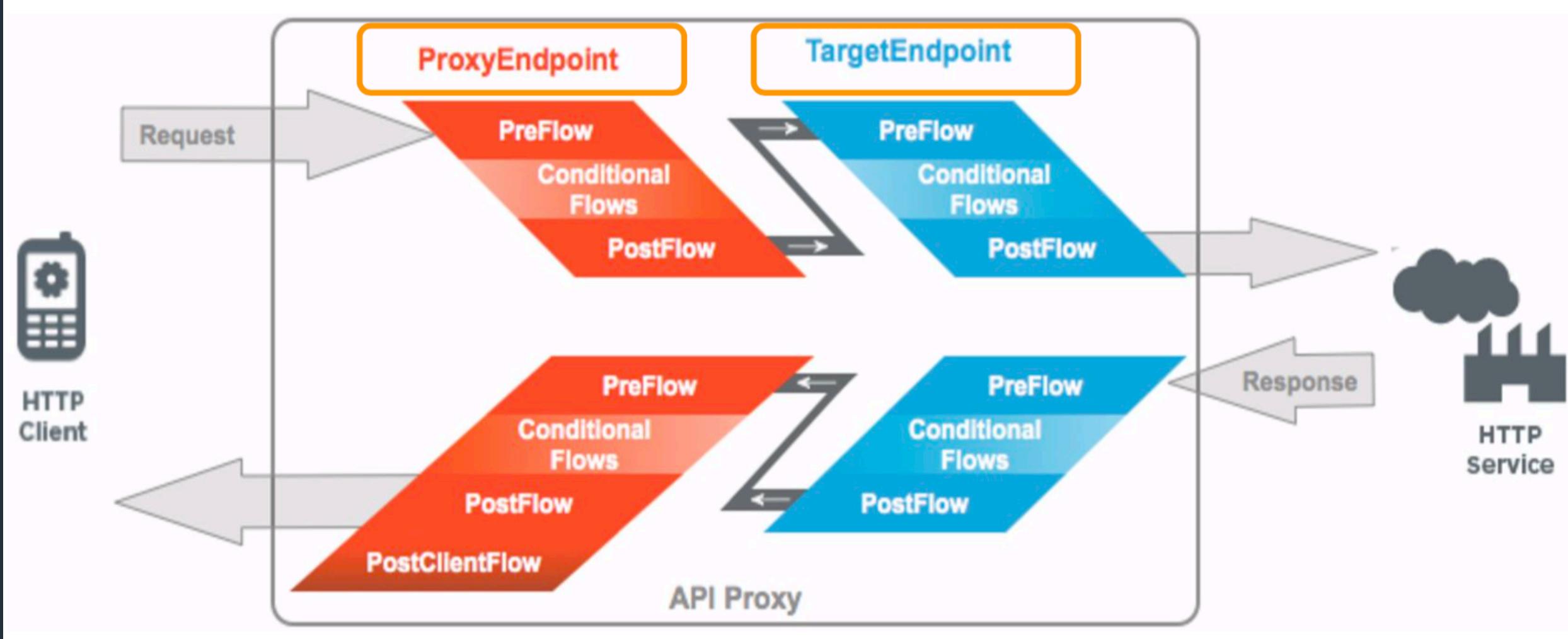
Apigee Edge Concepts



Apigee Edge Pipeline



Apigee Edge Proxy Flows



Apigee Edge RBAC

Org Admin	Business User	Operations Admin	User Developer
			
full access	analytics	deployments	write code

API Design

API Design

“An API is like a joke.
If you have to explain it, it’s
not that good.”

Martin LeBlanc, CIO Iconfinder

API Design tips

1. Invierta el 70% de su esfuerzo de diseño en API que tengan consumidores que puedan decirle si lo que está diseñando es útil (o no).
2. Pase más tiempo pensando en los recursos que se utilizan, no en las acciones que se toman. Los sustantivos son buenos; Los verbos son malos.
3. Antes de diseñar una nueva API, primero evalúe la cartera actual y vea si se puede hacer una modificación a una API existente para satisfacer la necesidad.
4. Los requerimientos no funcionales como los esquemas de seguridad, el SLA o el tiempo de latencia, si bien son importantes, no deberían afectar el diseño.
5. Evite la necesidad de implementar algo personalizado. En la mayor medida posible, utilice las convenciones y técnicas ya existentes.

Recursos orientados a sustantivos

1. Mantenga los recursos primarios en 2 niveles.
2. Use sustantivos plurales para colecciones.
3. Prefiere nombres concretos sobre abstracciones.

```
https://myserver/v1/dogs
```

```
https://myserver/v1/dogs/{dog-id}
```

Forzar verbos fuera de URI

Los verbos en el URI causan una larga lista de URI sin un patrón consistente.

```
/getAllDogs  
/verifyDogLocation  
/isFeedNeeded  
/giveDirectOrder  
  
/getDog  
/newDog  
/getNewDogsSince  
/getRedDogs  
/setDogStateTo  
/getAllLeashedDogs  
/createRecurringMedication
```

```
/getHungerLevel  
/getSquirrelChasingPuppies  
/newDogForOwner  
/transferDog  
/doDirectOwnerDiscipline  
/getRecurringFeedingSchedule  
/isDogSick  
  
/getDogsAtPark  
/feedADog  
/getSittingDogs  
/checkHealth
```

Use verbos HTTP para administrar operaciones CRUD

Resource	POST create	GET read	PUT update	DELETE delete
/dogs	Create a new dog	List matched dogs	Bulk update matched dogs	Delete all matched dogs
/dogs/1234	Error	Show "Fido"	If exists, update "Fido" If not, error!	Delete "Fido"

Product Strategies

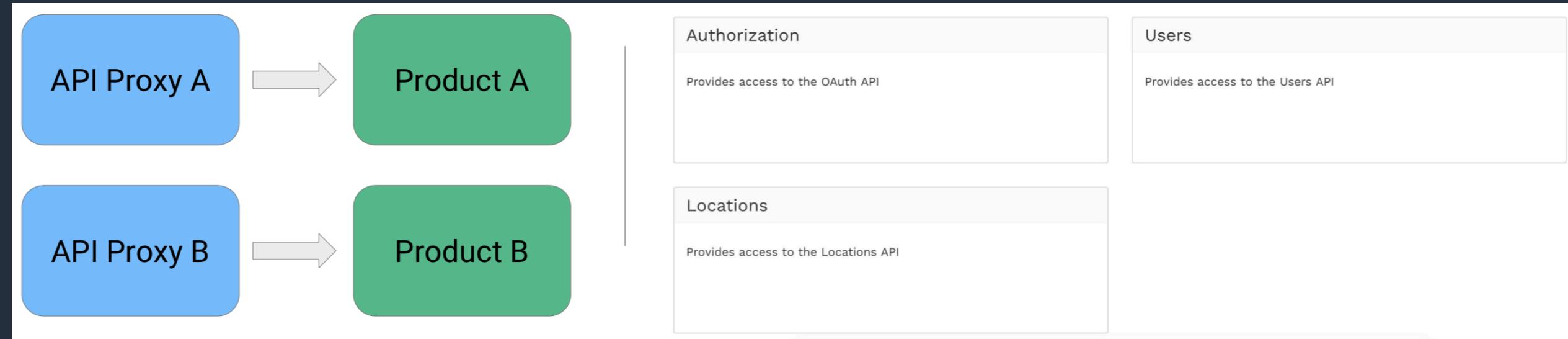
API Proxy Model

Service Plan Model

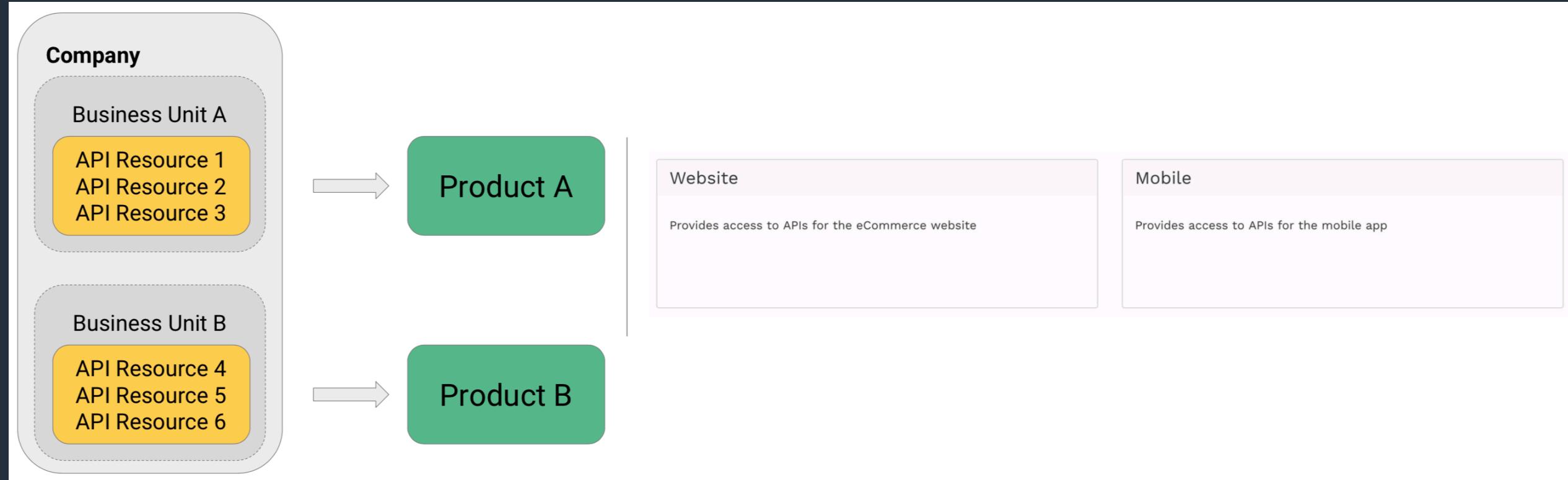
Business Model

Ownership Model

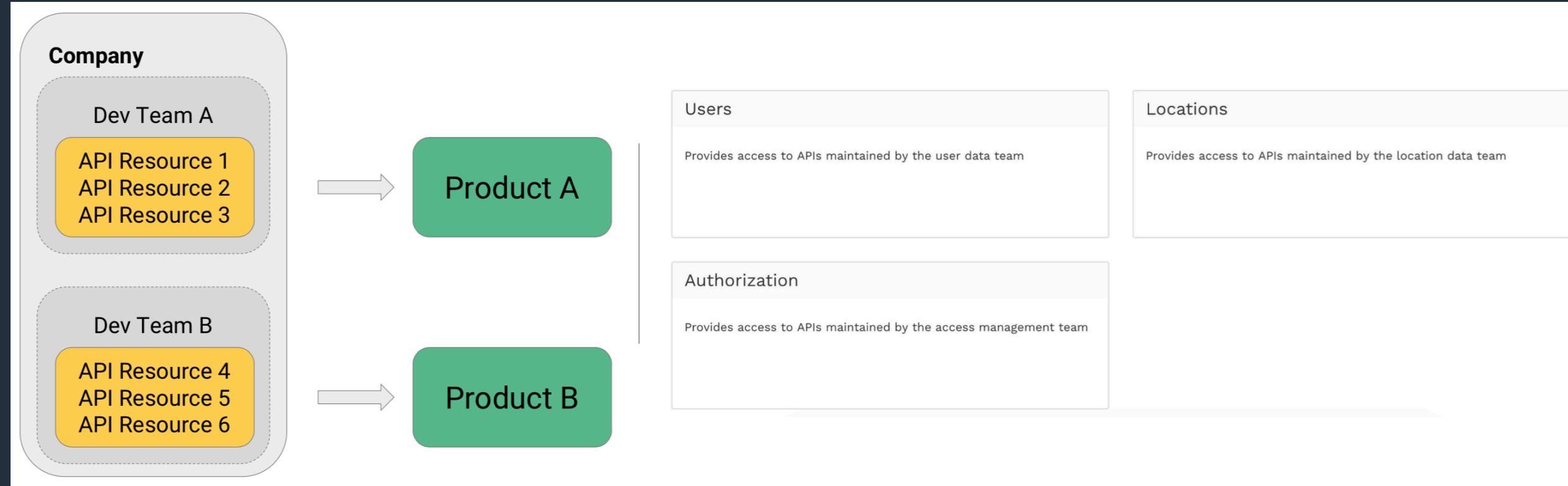
API Proxy Model



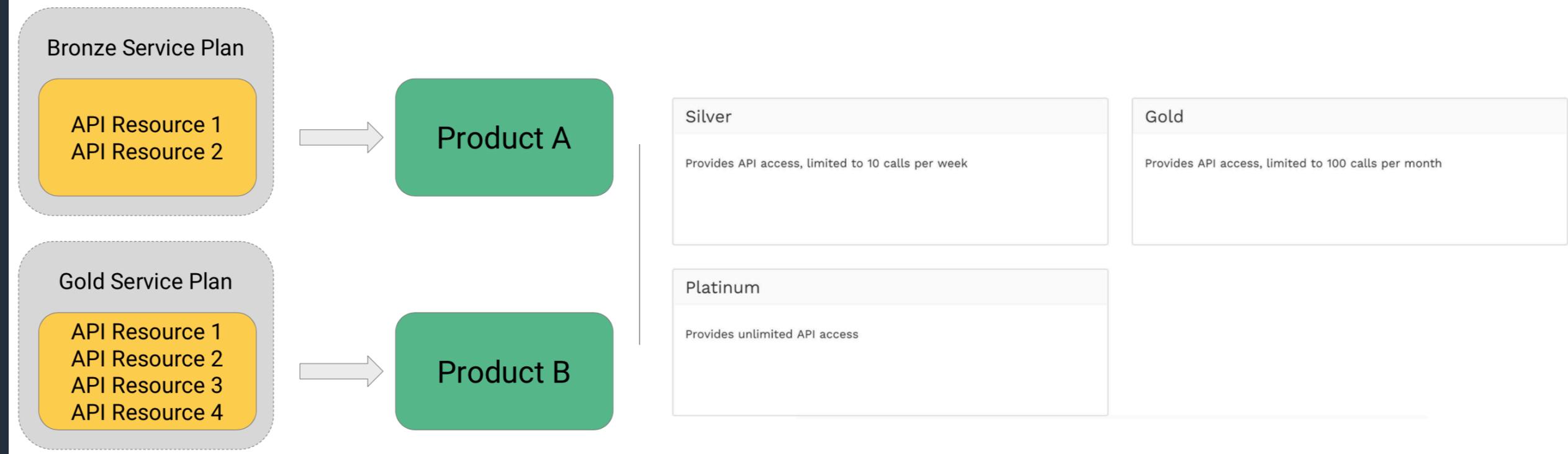
Business Model



Ownership Model



Service Plan Model





Muchas gracias