



CURSO ÚNICO EN MÈXICO

# ¡Bienvenido!

# Introducción a

# las APIs REST

**Jovani Arzate**

API Evangelist, Solution Architect Cloud, CTO de MegaCloud

Mail:[jovaniac@gmail.com](mailto:jovaniac@gmail.com)

Twitter: [@Jovani\\_Aarzate](https://twitter.com/Jovani_Aarzate)

Cel: 55-80975755

**apigee**



# Herramientas y tecnología

usadas en el curso:

- **Swagger Editor**
- **Cucumber Gherking**
- **Google Cloud Platform**
- **Docker-Hub**
- **Visual Studio Code**
- **Postman**
- **Node - NPM**



Google Cloud Platform

Swagger™



# Extras necesarios para el curso:

- Cuenta de Google cloud platform: Google Console - Tarjeta de crédito para recibir **\$300 USD gratis por 12 meses**
- Cuenta en Dockerhub: <https://hub.docker.com/>

# Logística del Curso

## **Requerimientos Live Virtual Class:**

- -Contar con conexión a internet.
- -Ingresar por navegador Google Chrome.
- -Ingresar al link de acceso para poder incorporar a la conferencia a partir del día de hoy.
- -Apagar tu cámara y poner en mute tu micrófono (mejora la transmisión por el uso elevado en datos de la cámara).
- -Cuando deseas participar desactiva el mute de tu micrófono y pide la palabra al instructor.
- -Te enviaremos un mensaje con el link de acceso, además de un correo con la invitación al Bootcamp donde están los datos para el ingreso al LVC .

## **Importante:**

- El link será el mismo en todas las sesiones.

## **Importante:**

- Las sesiones LVC son únicas y no quedan grabadas por lo que te recomendamos estar atento a la clase y resolver las dudas con el instructor al instante. Si tienes alguna duda o comentario puedes contactarnos de manera directa en whatsapp o correo electrónico.

## **Breaks y Lunch**

- Break de 10 minutos a las 8:00 PM



# Vacantes nuevas respecto al curso

- **API Developer**
- **API Architect**



# Experiencia sobre las tecnologías del training

- APIs
- Apigee
- Microservicios



**¿Qué ha hecho tu  
instructor  
durante los  
últimos 3 años?**

# Presentación

- Nombre
- Trabajo actual y rol
- Experiencia en APIs, Apigee, microservicios

# Logística del Curso

Repositorio de trabajo

Clonar repositorio

1.- git clone

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training\\_apis\\_apigee\\_enero\\_2024](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training_apis_apigee_enero_2024)

Bajar actualizaciones, tener cuidado en no subir features o cambios

2.- git pull

3.- Las credenciales son las mismas para todos.

# Logística del Curso

## Estructura de material



01\_openapi



02\_apigee



03\_materia\_oficial



04\_ofertas\_trabajo



06\_colecciones\_postman



07\_books



general.txt

# Módulo I

## APIs

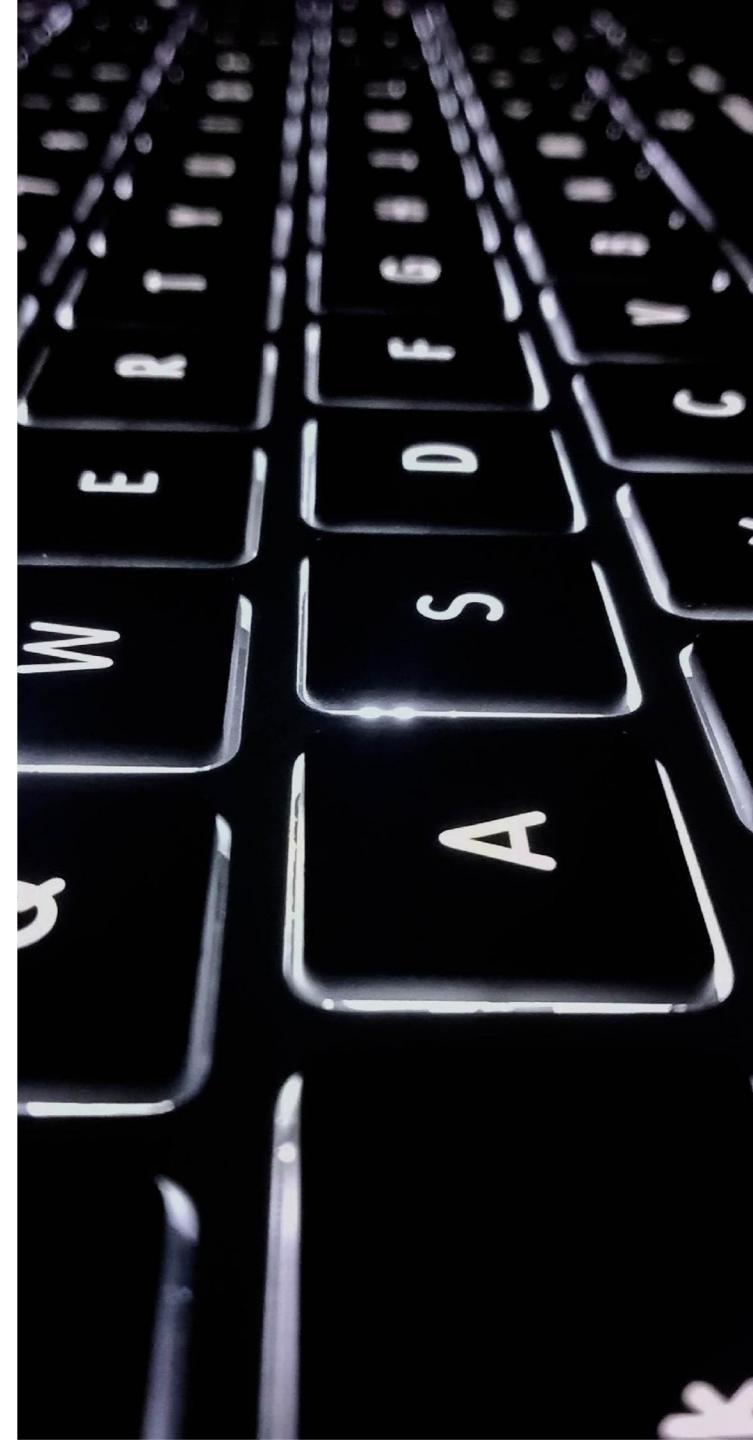
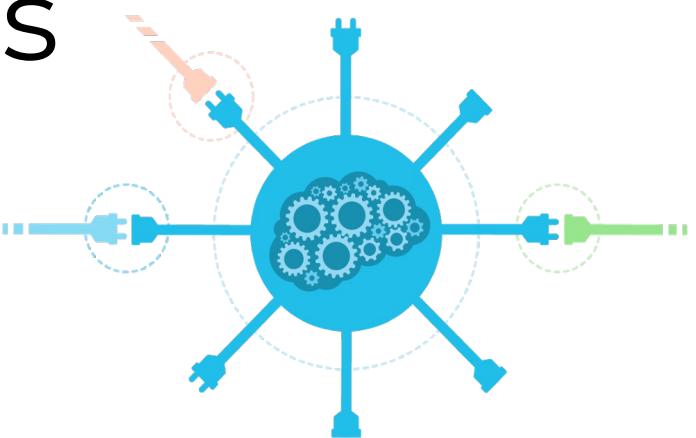


### TEMARIO

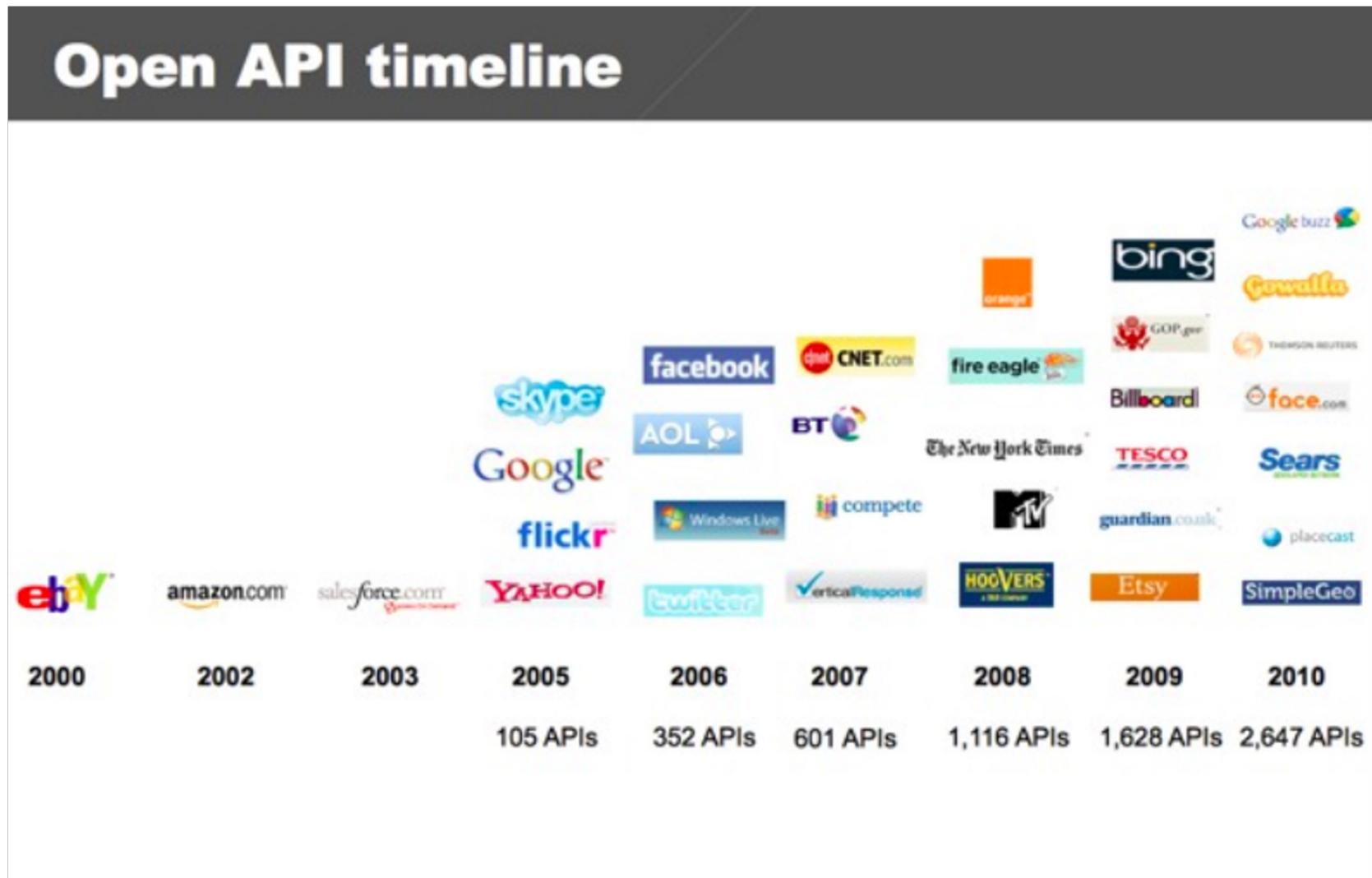
1. APIs - APIficando productos digitales
2. APIs
3. Estrategias de APIs
4. Transformación digital
5. Tipos de APIs
6. Dominios de APIS
7. OpenAPI 2.0 y 3.0  
Swagger 2.0, OpenAPIs 3
1. API Design
2. Contract First
3. API First Development
4. REST Estilo
12. API Development
13. Diseño de productos Digitales
14. Estrategia de Mocks
15. BDD Behavior Driven Development
16. Cucumber
17. APIs con Docker y Kubernetes

# Módulo I

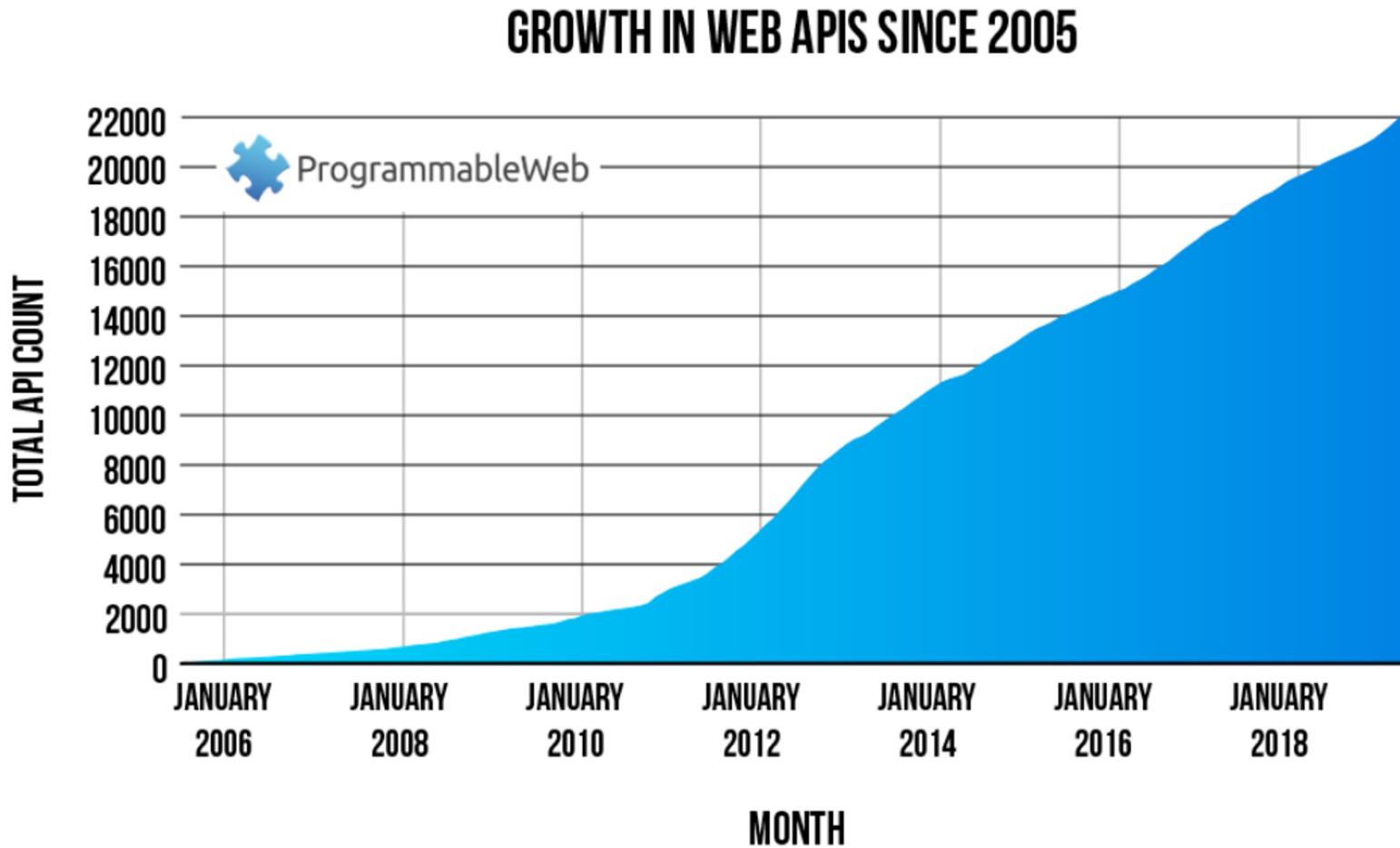
## APIs – APIficando los recursos de las empresas



# OPENAPI - Time Line

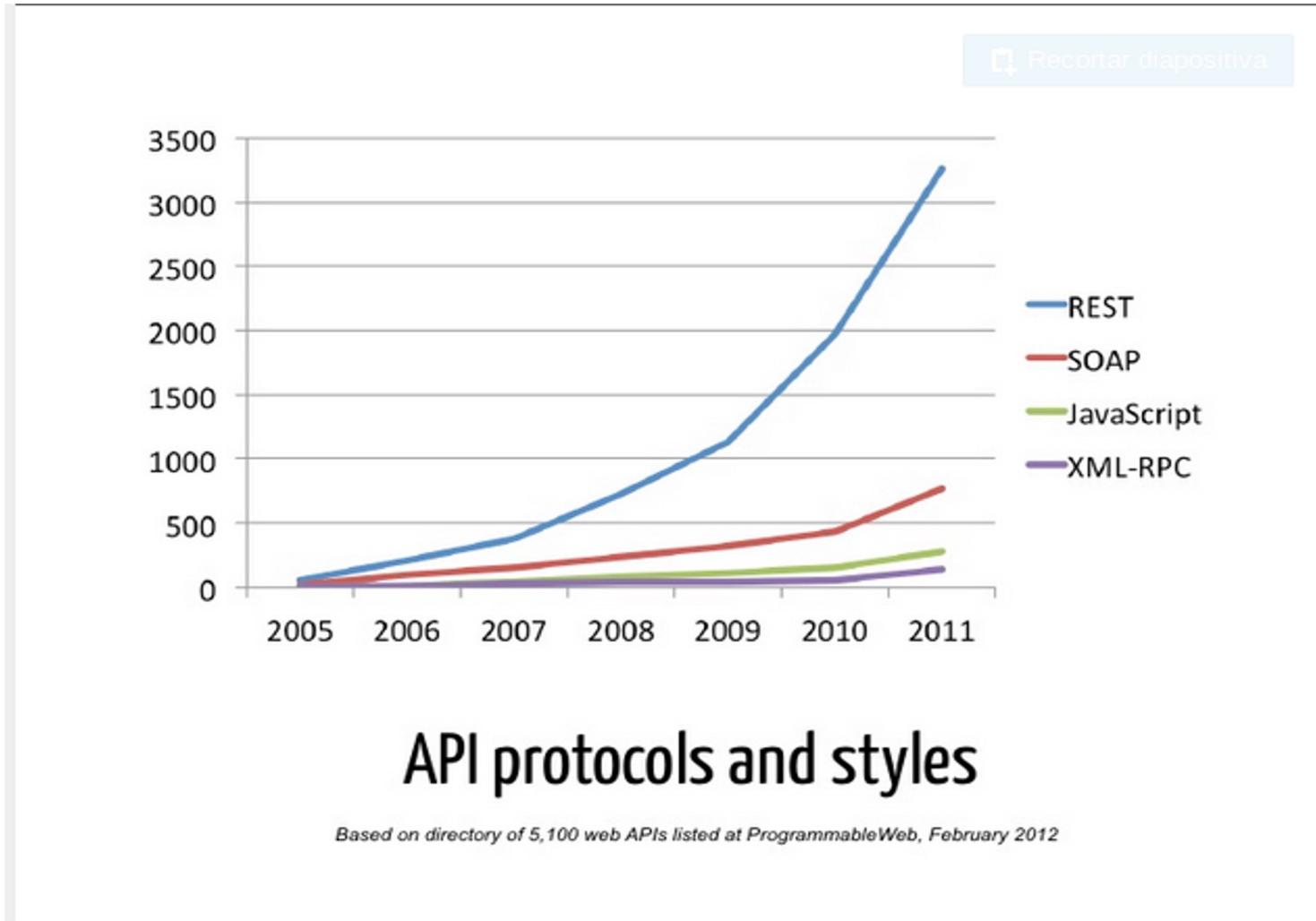


# OPENAPI - Time Line



The growth over time of the ProgrammableWeb API directory to more than 22,000 entries

# OPENAPI – Protocolos y estilos



# OPENAPI – API Club

## API Billionaires Club

- twitter** 13 billion API calls / day (May 2011)
- Google** 5 billion API calls / day (April 2010)
- facebook** 5 billion API calls / day (October 2009)
- NETFLIX** 1.4 billion API calls / day (May 2012)
- ACCUWEATHER** 1.1 billion API calls / day (April 2011)
- KLOUT** 1 billion API calls / day (May 2012)
- eBay** 1 billion API calls / day (Q1 2012)
- Sabre** 1 billion API calls / day (January 2012)

# OPENAPI – APIs como productos

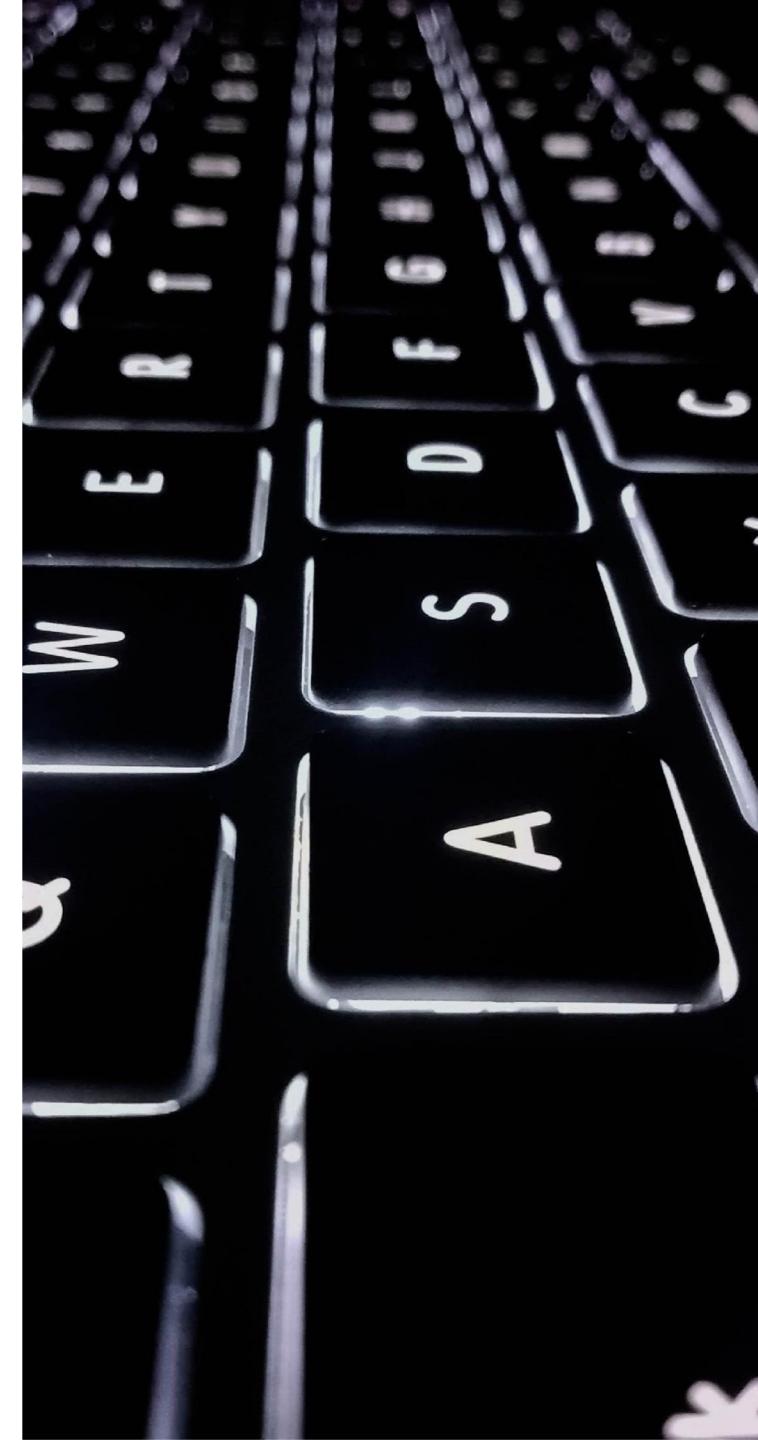
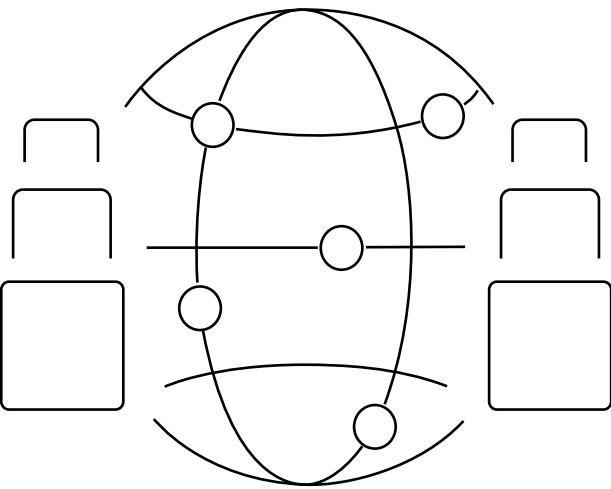
## API as Product



customer == developer

# Módulo I

## APIficando los sistemas de las empresas



# ¿Qué son las APIs?

Una API es una interfaz que facilita que una aplicación "**consume**" capacidades o datos de otra aplicación.

Al definir puntos de entrada estables y simplificados para la lógica y los datos de la misma aplicación.

Las APIs permiten a los desarrolladores **acceder y reutilizar fácilmente la lógica de la aplicación** creada por otros desarrolladores.

En el caso de las 'APIs web', esa lógica y los datos se exponen a través de la red.



# Siempre APIs...

Siempre hemos usado APIs

## Web service APIs

- SOAP
- XML-RPC and JSON-RPC
- REST

## WebSocket APIs

## Library-based APIs

- JavaScript
- TWAIN

## Class-based APIs (object orientation)

- Java API
- Android API

## OS functions and routines

- Access to file system
- Access to user interface

## Object remoting APIs

- CORBA
- .NET Remoting

## Hardware APIs

- Video acceleration
- Hard disk drives
- PCI buses

# Oportunidades de negocio

En la economía digital, las APIs son esenciales para **ejecutar ideas con rapidez** y aprovechar nuevas oportunidades de negocio.

# Estrategia empresarial con APIs

Con un aumento exponencial en la cantidad de aplicaciones requeridas en el mundo digital, las empresas **necesitan exponer más datos a través de las APIs**, para proporcionar experiencias de aplicaciones más ricas y personalizadas para usuarios internos, clientes y socios.



# Estrategia empresarial con APIs

bbvapimarket.com/home

Contacto FAQs English

PRODUCTOS API CÓMO FUNCIONA NIVELES DE ACCESO RECURSOS

## APIs de BBVA

Regístrate ahora CONOCE API\_MARKET

¿Quieres saber cómo integrar la APIs de BBVA en tu empresa? Contacta con nosotros

API\_Market es una plataforma global y abierta de APIs que te permite acceder a soluciones financieras de manera ágil e implementarlas fácilmente en tu empresa.

Puedes gestionar, controlar y analizar pagos, verificar la identidad y notificar a tus clientes, acceder a patrones de compra segmentados y mucho más.

IBM API Connect /dev

Home Getting Started API Products

Domain Channel Mgmt 1.0.1

CHNN-BNE-V-Transactions V1.1.0 Open API

APIs APIFactory OAuth Provider 1.0.0

CHNN-BNE-V-LoginCount 1.0.0

Accounts Search 1.0.0

CHNN-BNE-V-MigrationStatistics 1.0.0

CHNN-BNE-V-DBMigration 1.2.0

CHNN-BNE-V-FileMigration 1.1.0

CHNN-BNE-V-AccountBalance 1.1

CHNN-BNE-V-Summary

CHNN-BNE-V- https://sit.developer.banamex.com

CHNN-BNE-V-Transactions V1.1.0

No votes yet

Description Service for getting Pending Authorization Transactions

Endpoints https://sit.api.banamex.com/mx-gcgo/pst PRODUCTION DEVELOPMENT

Paths /v1/channels/bne/accounts/transactions/retrieve

POST /v1/channels/bne/accounts/transactions/retrieve pendingAuthorization

Example Request

```
curl -request POST \ -url https://sit.api.banamex.com/mx-gcgo/sections/retrieve \ -header "accept: REPLACE THIS VALUE"
```



círculo de crédito API Hub Guía de inicio APIs Noticias Contacto

Detonamos el potencial de innovación

API Hub es el programa de gestión de APIs de Círculo de Crédito. Creemos que la innovación es impulsada por gente con creatividad y determinación que desea impactar el mundo de forma positiva.

Nuestra plataforma proporciona información estratégica a los desarrolladores independientes y empresas que están creando aplicaciones innovadoras para ayudar a las personas a cumplir sus sueños.

https://www.bancoazteca.com.mx/empresas/apilab.html

Banco Azteca

PERSONAS EMPRESAS

API Lab

Bienvenido a API Lab

JEl lugar donde tendrás un nuevo modelo de negocio digital!

¡Di sí! ¡Abre tus posibilidades!

REGÍSTRATE

adn  
40

Te invitan al foro:  
"Las APIs en la transformación de México"  
En el marco del lanzamiento de  
APILab de Banco Azteca

# Estrategia empresarial con APIs

Ejemplo: BBVA API Market

[https://www.youtube.com/watch?v=8IxUDAs6c9Y&feature=emb\\_title](https://www.youtube.com/watch?v=8IxUDAs6c9Y&feature=emb_title)

# Open banking

El open banking o banca abierta, es un sistema ajeno a los bancos que **permite manejar más fácilmente el dinero.**

Es un sistema que proporciona a un usuario una red de datos de las instituciones financieras a través del uso de interfaces de programación de aplicaciones (APIs), con el fin de **mejorar la experiencia bancaria de los clientes y promover la competencia** entre grandes y pequeños bancos.

*Fomenta la competencia, pero es difícil lograrla en equilibrio por la diferencia de alcance en clientes.*



# Impulsores de la transformación digital



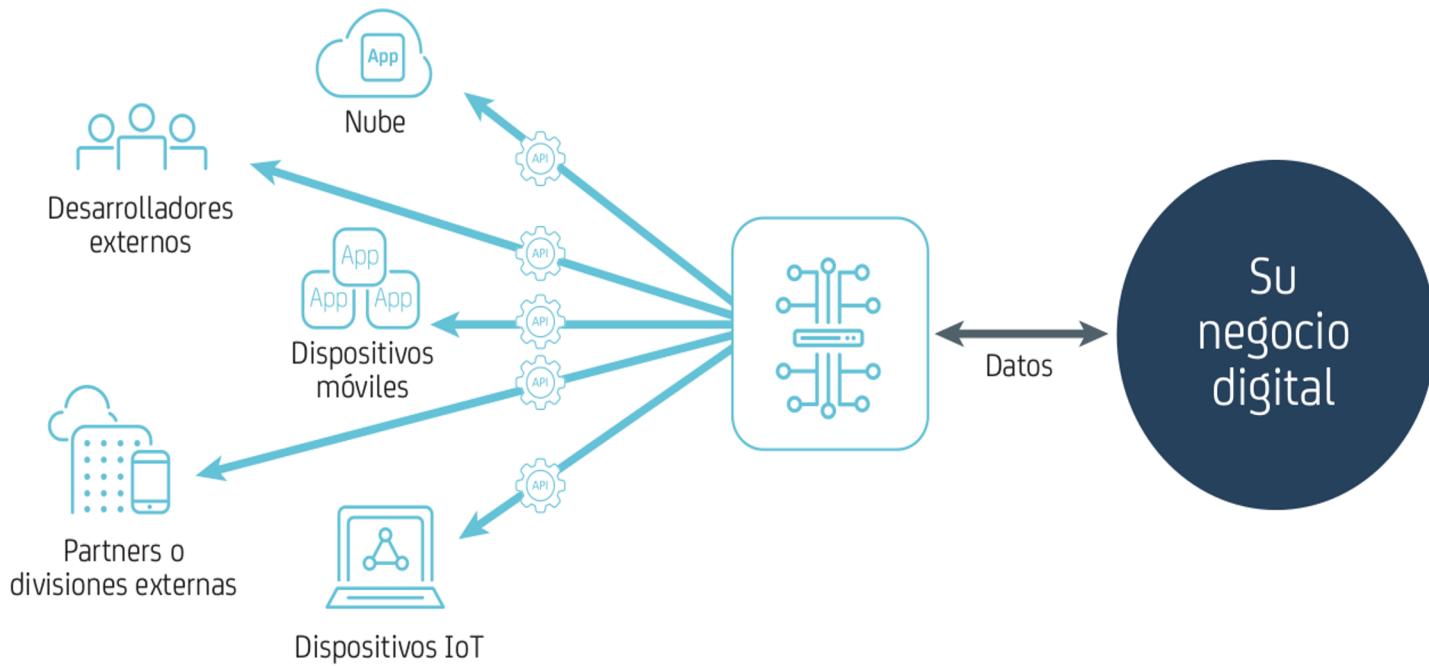
<sup>1</sup> Freeform Dynamics, Exploiting the Software Advantage; Lessons from Digital Disrupters, octubre de 2015

# Iniciativas digitales importantes

1. **Optimizar la integración** de los sistemas internos y de los partners.
2. **Facilitar y compartir los datos** con los partners para capturar nuevas oportunidades de mercado.
3. **Ampliar la interacción** con el cliente por medio del Internet de las cosas.
4. **Proporcionar acceso multicanal** a los clientes.
5. **Acelerar** el proceso de **desarrollo** de aplicaciones móviles.



# Negocio digital



En la economía digital, las APIs son esenciales para **ejecutar ideas con mayor rapidez** y aprovechar las nuevas oportunidades de negocio.

Son los pilares de la transformación digital, ya que, permiten

1. Ofrecer **experiencias excepcionales** a los clientes
2. **Flujos de ingresos** nuevos
3. **Conexión** con los empleados, socios y aplicaciones
4. **Dispositivos** con los datos en cualquier momento y lugar.

# Reutilización y valor

## APIs

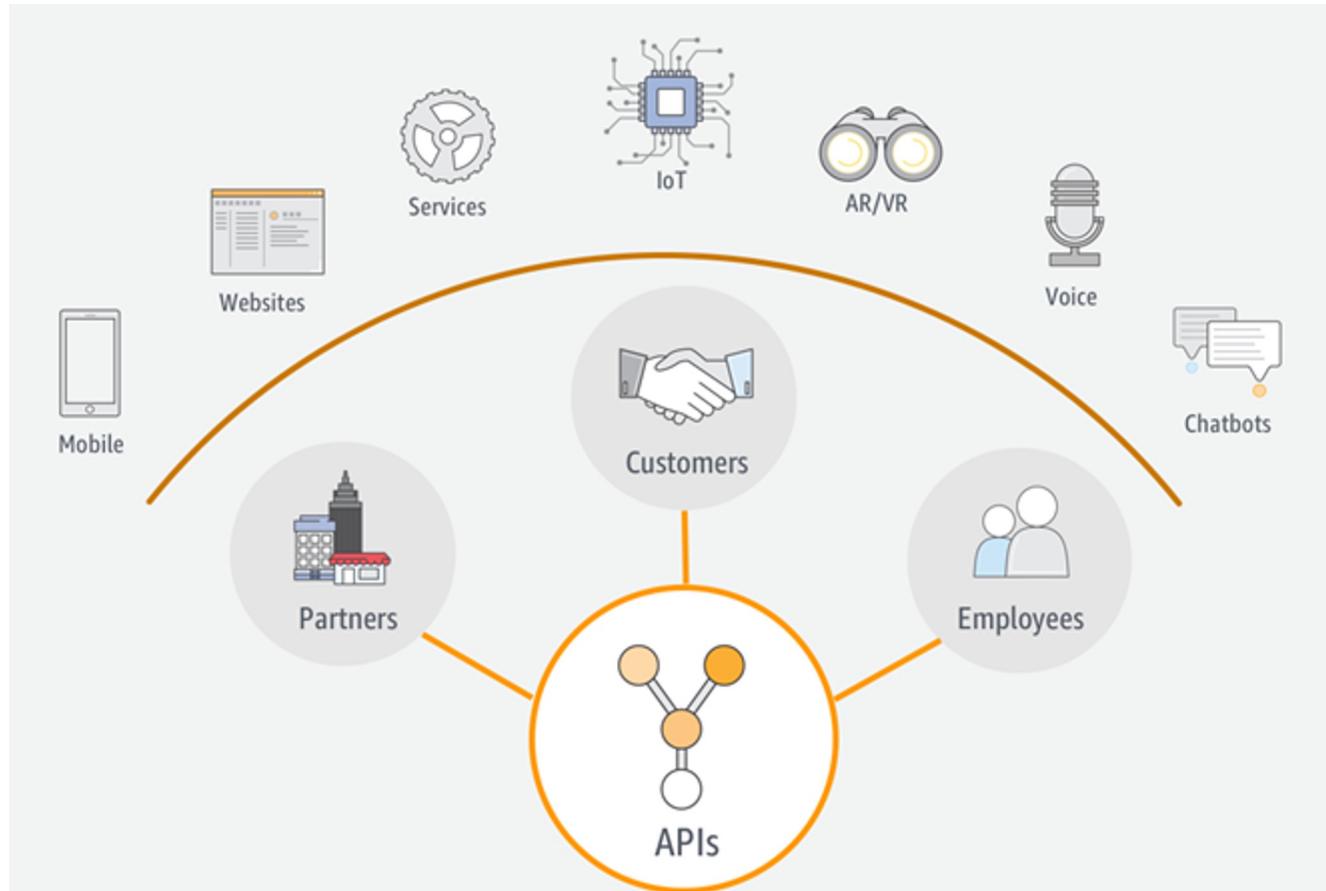
Ayudan a la **reutilización de sistemas** con un gran valor organizacional en aplicaciones que tienen más probabilidades de ofrecer un valor empresarial directo.

Existen para **ofrecer un valor de negocio** (pero sin ofrecer eficiencia técnica), lo hacen de un modo más indirecto que la Web basada en exploradores, donde un sitio puede ofrecer ventas o captar clientes.

Generan ingresos de un modo más sutil, a través de la conexión de diferentes activos.

## VALOR EN LAS APIs

1. Ingresos nuevos en forma directa
2. Ampliación del alcance al cliente (Dispositivos)
3. Marketing y ventas
4. Posibilita la innovación empresarial y técnica (backends)



# Tipos de APIs

## Servicio Web

El tipo de diseño de servicio web es un enfoque de diseño de API basado en operaciones e independiente del transporte que emplea el lenguaje de descripción de servicios web (WSDL) para describir interfaces.

Su origen se encuentra en el ámbito SOA.

## Rest práctica

Estilo que emplea URI en lugar de WSDL. Está vinculado al transporte (únicamente admite HTTP), ha superado con creces el diseño de servicio web en el diseño de APIs empresariales.

El término “API web” se emplea habitualmente como sinónimo de “API RESTful” y alcanza el estado de “RESTfulness” considerando un objetivo clave de cualquier proyecto de diseño de interfaces.



# Tipos de APIs

## Hipermedia (HATEOAS)

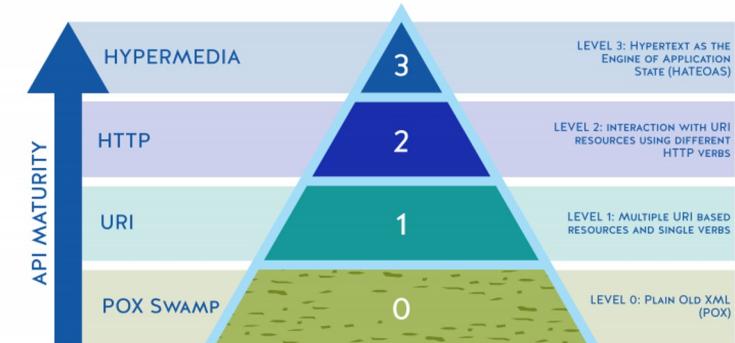
Un cliente interacciona con una aplicación de red cuyos servidores de aplicación proporcionan información dinámicamente a través de hipermedia.

## ASYNCAPI

AsyncAPI es una especificación que permite definir APIs y que se enfoca en arquitecturas orientadas en eventos (EDA).

Una arquitectura orientada a eventos es un paradigma de diseño en el que una pieza de software (un consumidor de eventos) recibe una notificación de que un evento o mensaje ha sido publicado en un canal por otra pieza de software (un generador de eventos, o publicador).

THE RICHARDSON MATURITY MODEL



NORDICAPIS.COM

GET /accounts/12345

HTTP/1.1 200 OK  
Content-Type: application/vnd.acme.account+json  
Content-Length: ...

```
{  
  "account": {  
    "account_number": 12345,  
    "balance": {  
      "currency": "usd",  
      "value": 100.00  
    },  
    "links": {  
      "deposit": "/accounts/12345/deposit",  
      "withdraw": "/accounts/12345/withdraw",  
      "transfer": "/accounts/12345/transfer",  
      "close": "/accounts/12345/close"  
    }  
  }  
}
```



AsyncAPI

# APIs internas

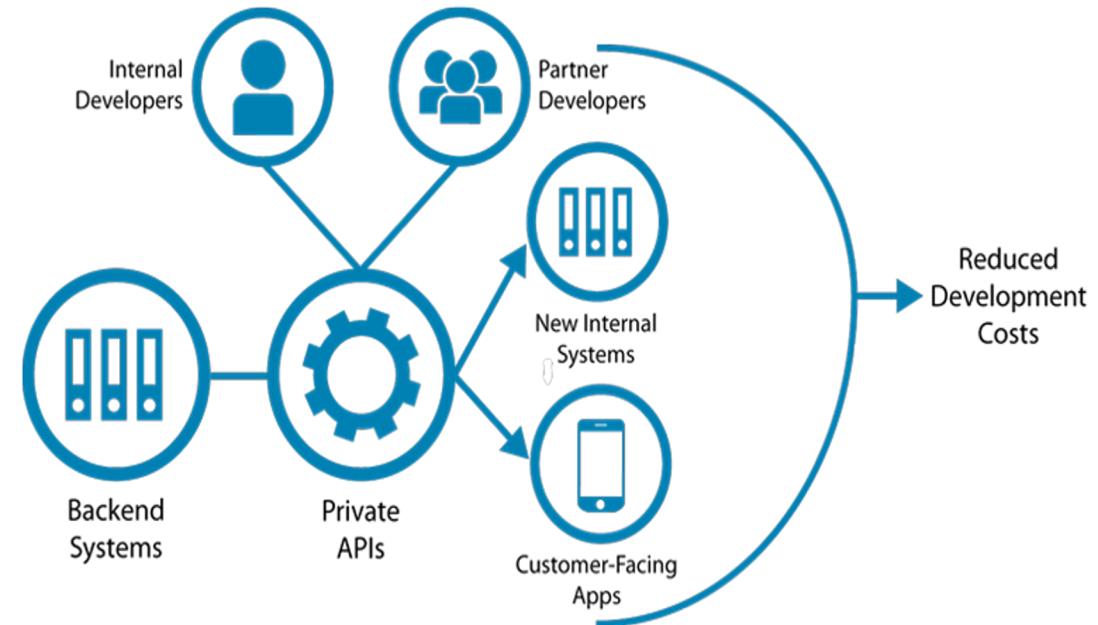
Una API interna o privada es una interfaz que abre partes de los datos de back-end de una organización y la funcionalidad de la aplicación para uso de los desarrolladores que trabajan dentro de la organización.

Las nuevas aplicaciones pueden distribuirse públicamente, pero **la interfaz no está disponible para personas externas** de la empresa.

Las API privadas pueden reducir significativamente:

1. **Tiempo** de desarrollo
2. **Recursos** para integrar sistemas de TI internos
3. Crear **nuevos sistemas** que maximicen la productividad
4. Crear **aplicaciones orientadas al cliente** que amplíen el alcance del mercado y agreguen valor a las ofertas existentes.

*En lugar de crear aplicaciones aisladas desde cero, los desarrolladores pueden aprovechar un conjunto común de activos de software internos.*



# APIs externas

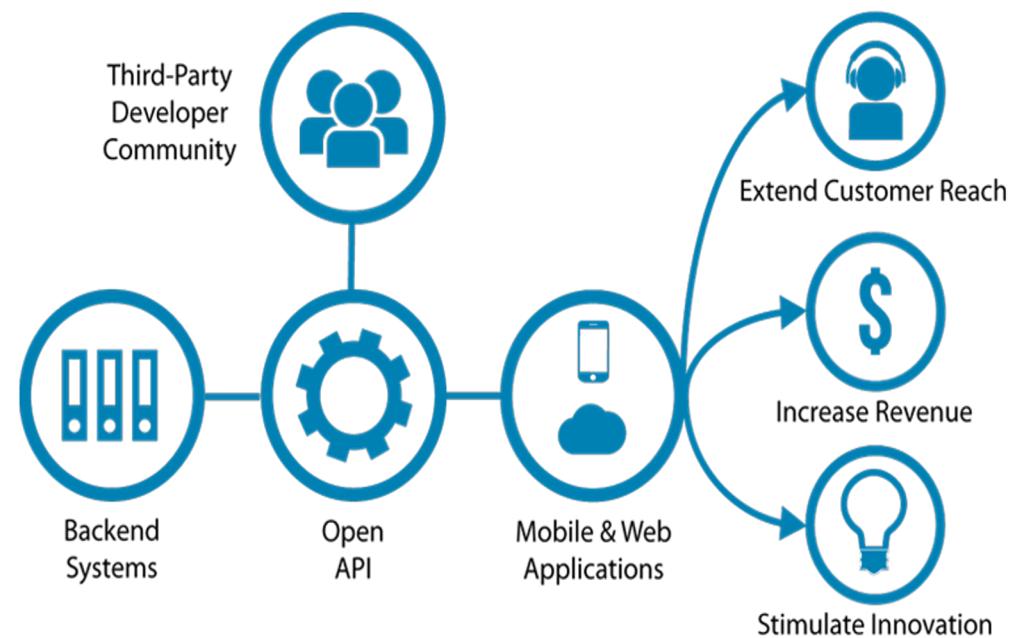
Una API externa es una interfaz diseñada para que la mayoría de desarrolladores web y móviles puedan acceder fácilmente al exterior.

Esto significa que los desarrolladores dentro de la organización que publicaron la API pueden usar una API abierta o cualquier desarrollador fuera de esta que se registre para acceder a la interfaz.

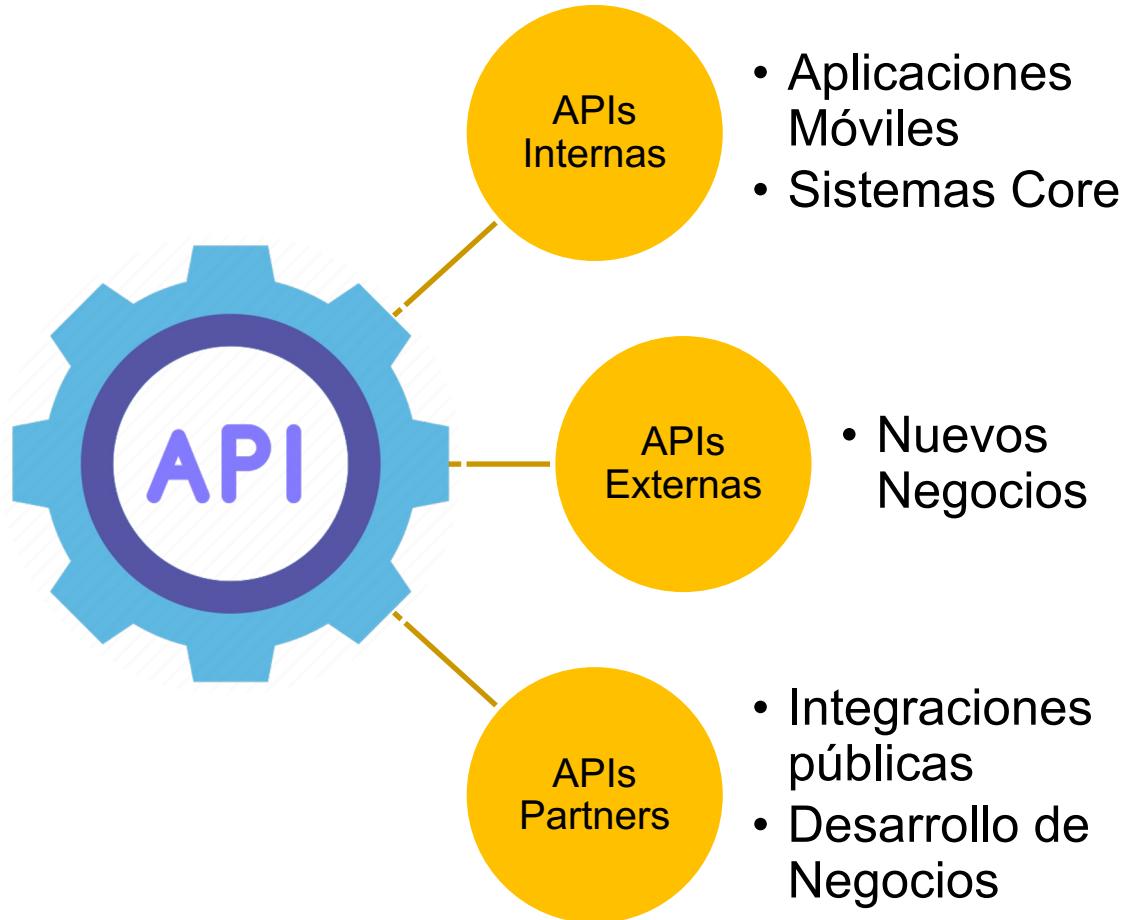
Esto permite que:

1. La organización **estimule el desarrollo de aplicaciones** innovadoras.
2. **Agregue valor a su negocio** principal, sin invertir directamente en los esfuerzos de desarrollo.
3. Y simultáneamente, **aumente la producción de nuevas ideas**, disminuyendo costos de desarrollo.

*Los desarrolladores internos pueden usar una API abierta, pero es justo decir que, en la mayoría de los casos, el éxito de un programa API abierto dependerá de su capacidad para atraer desarrolladores externos y ayudarlos a crear nuevas aplicaciones valiosas que la gente quiera usar.*



# Tipos de APIs



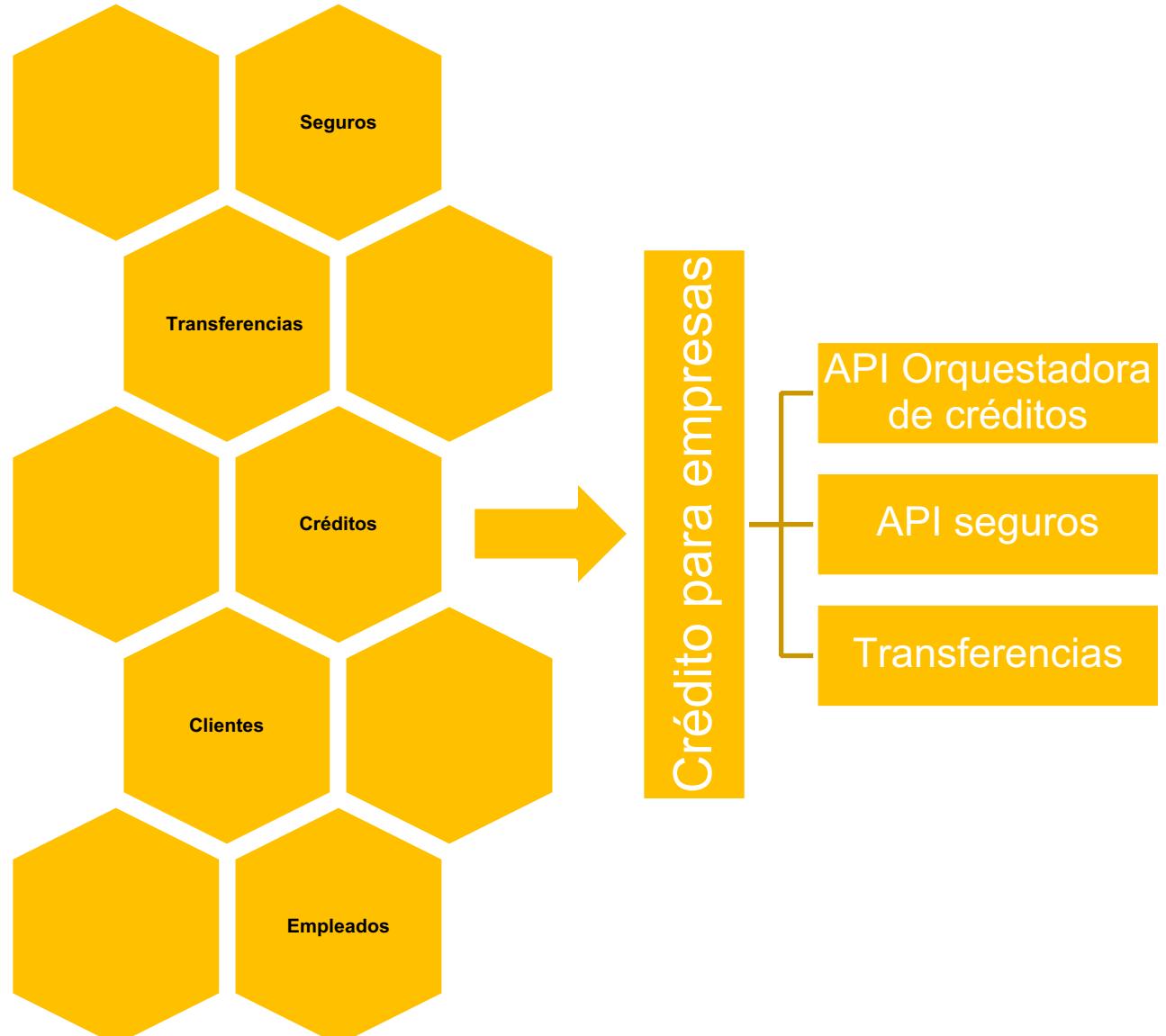
# APIs, dominios y productos

Las APIs de **dominio** son todas las **funcionalidades específicas** core dentro de un sistema de software en forma de servicio.

APIs de producto son la orquestación de un proceso de negocio que ofrece una **funcionalidad completa**.

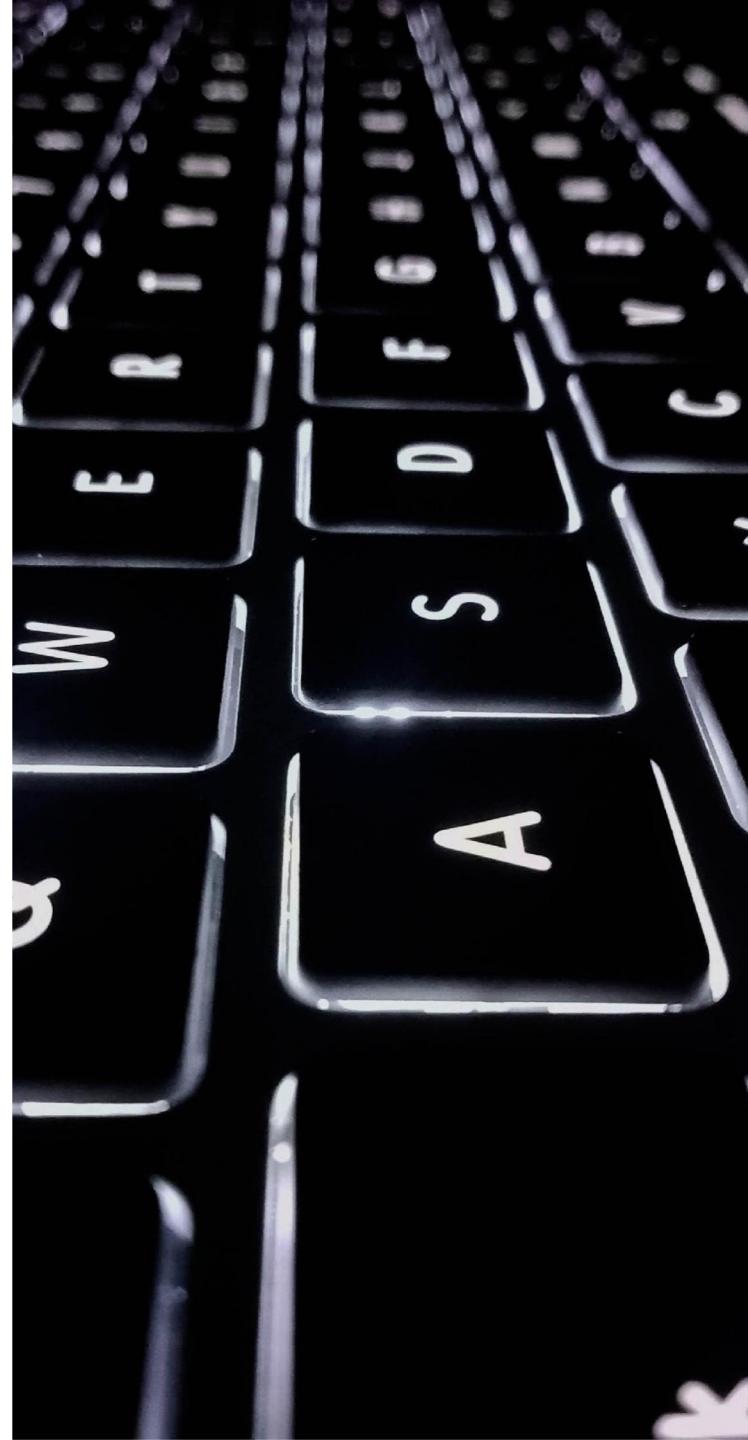
Podemos reutilizar diferentes APIs de dominio para poder generar todo un flujo de interacción y orquestación que ofrecerá experiencia de un proceso de negocio como “**producto**”.

- APIs Orquestadores
- APIs de Dominio
- APIs de Entidad
- APIs de Utilidad



# Módulo I

## Contract First / API First

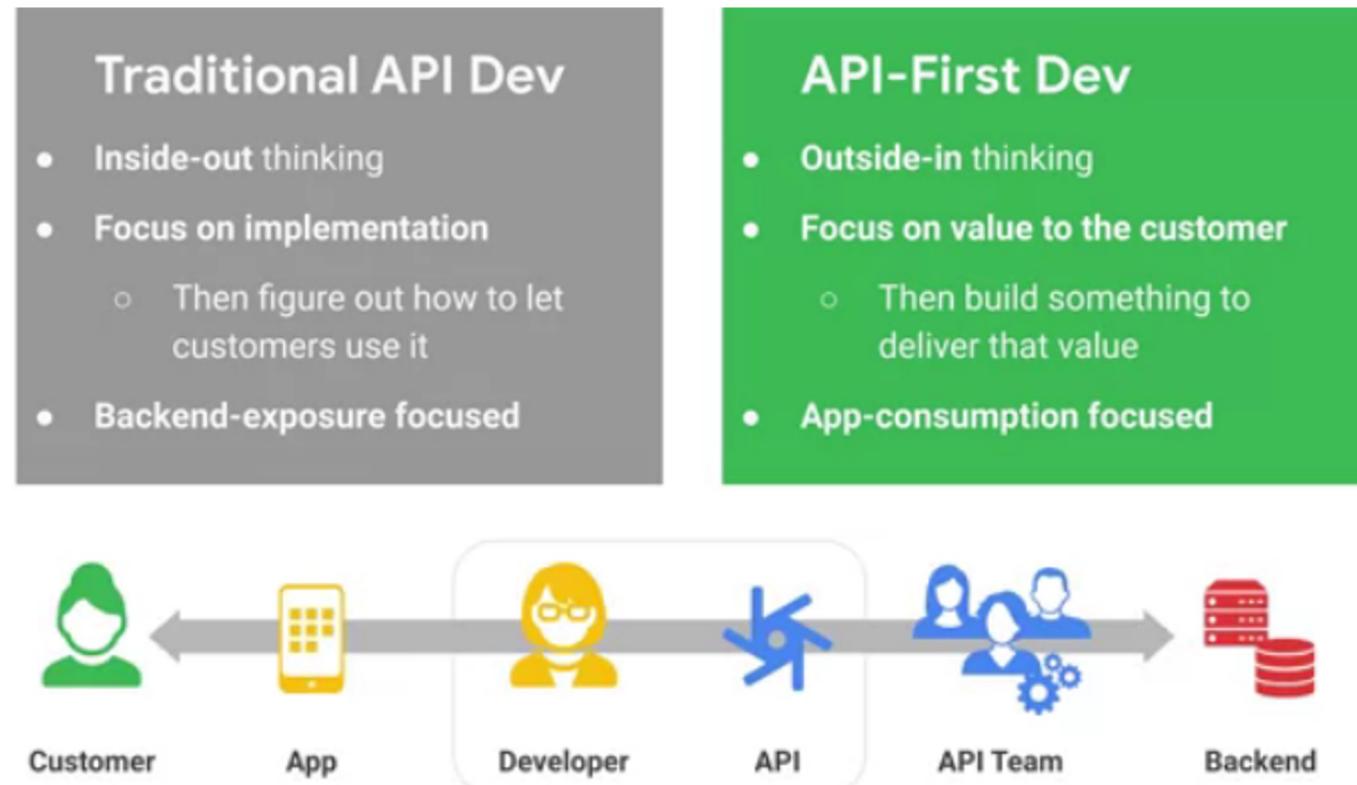


# Construcción de Productos

Todos los productos nuevos se deben construir o implementar como **API-First**.

**Esto significa construir la capa de integración y el servicio temprano en el proceso de desarrollo.** Eliminar el acoplamiento estrecho entre nuestras interfaces y nuestros datos para permitir:

- Ciclos de liberación más rápidos y reductores de bloqueadores durante el desarrollo.
- Una aplicación fácilmente flexible que puede adaptarse rápidamente a plataformas e implementaciones adicionales.
- Despliegue rápido de nuevos productos sin plazos de desarrollo prolongados.
- Compartir trabajo entre unidades de negocio.



# Contract First

Es una estrategia en el diseño de APIs, la cual establece en primero, **diseñar la API** (como tal, la especificación).

Esto implica que la **primera interfaz de su aplicación futura es la API**.

Sus futuros **usuarios-clientes** desarrollarán contra la API, por ellos, debemos construirlas teniéndolos en cuenta.

## API de Clientes del banco PODER 0.0.1

[ Base URL: [www.bancopoder.com/clientes/v1](http://www.bancopoder.com/clientes/v1) ]

API para poder realizar la captacion de nuevos clientes del banco poder

Schemes

HTTPS

Authorize



default

POST

/ creacion de nuevos clientes



GET

/ Busqueda de clientes



PUT

/ Actualizacion de clientes



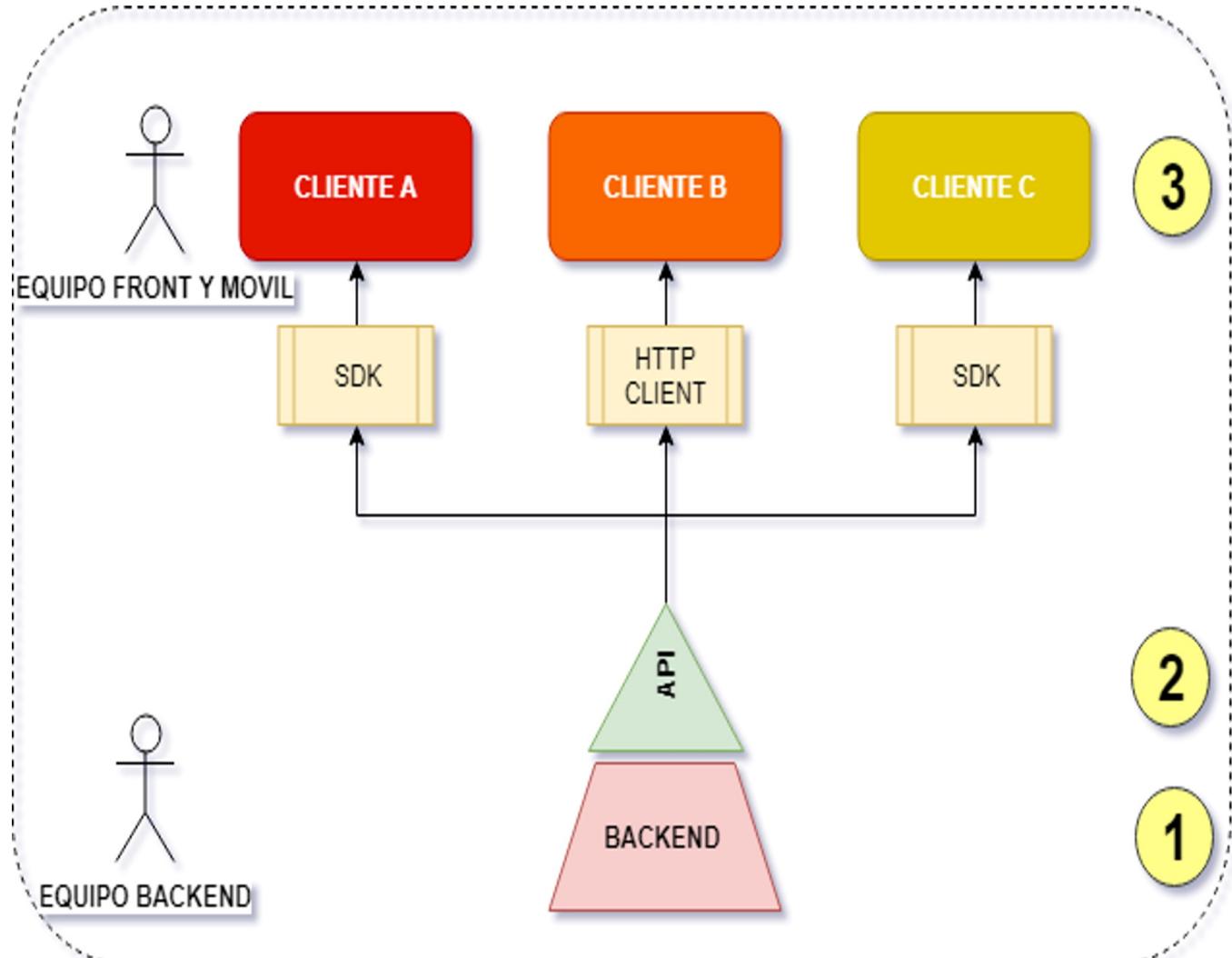
DELETE

/ Baja de clientes

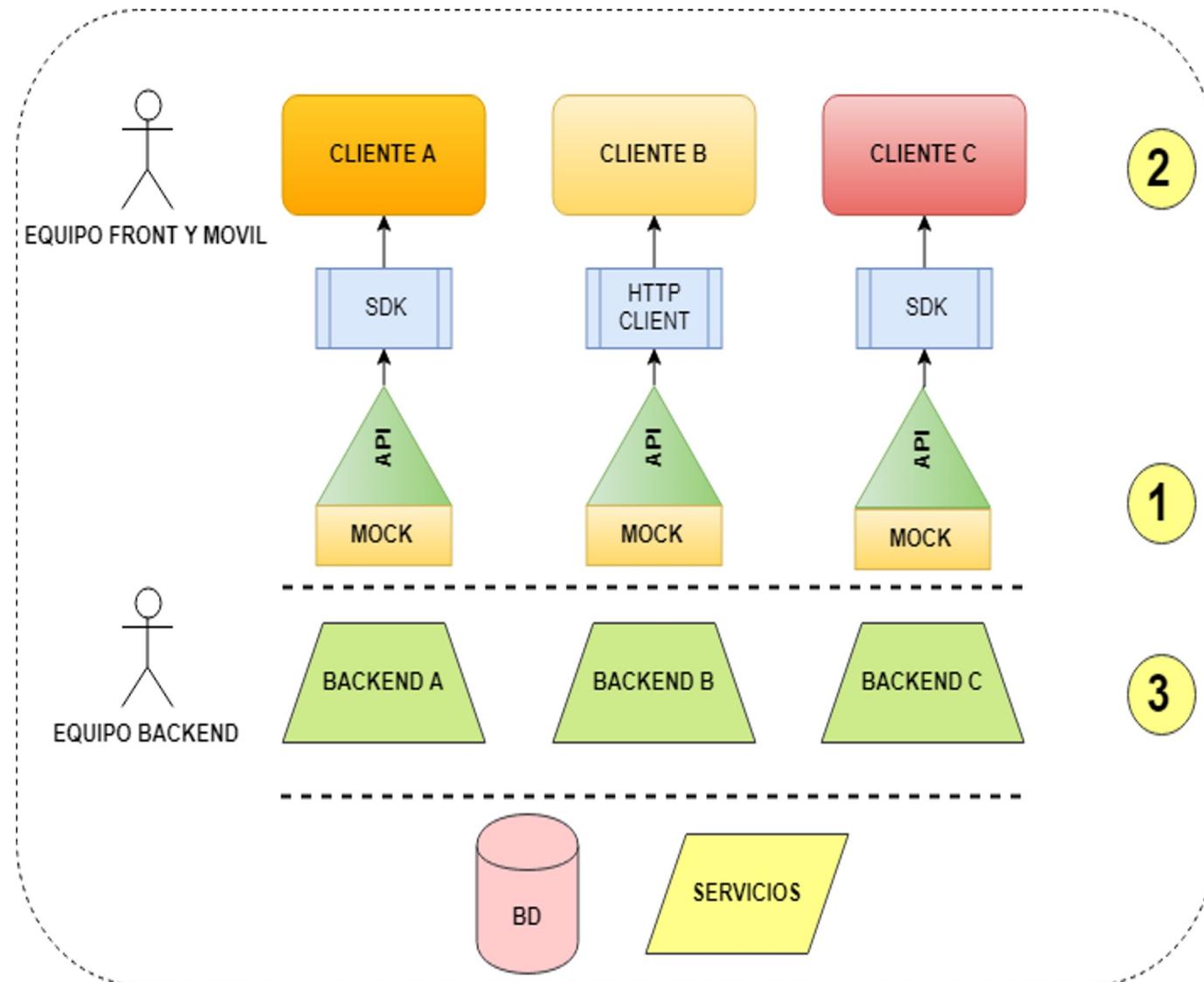


# API First

API-first development es una estrategia en la que la primera tarea consiste en desarrollar una API, que ponga en primer lugar los **intereses de su desarrollador**, la exposición objetivo y posteriormente, la construcción del producto sobre ella.



# API First Development



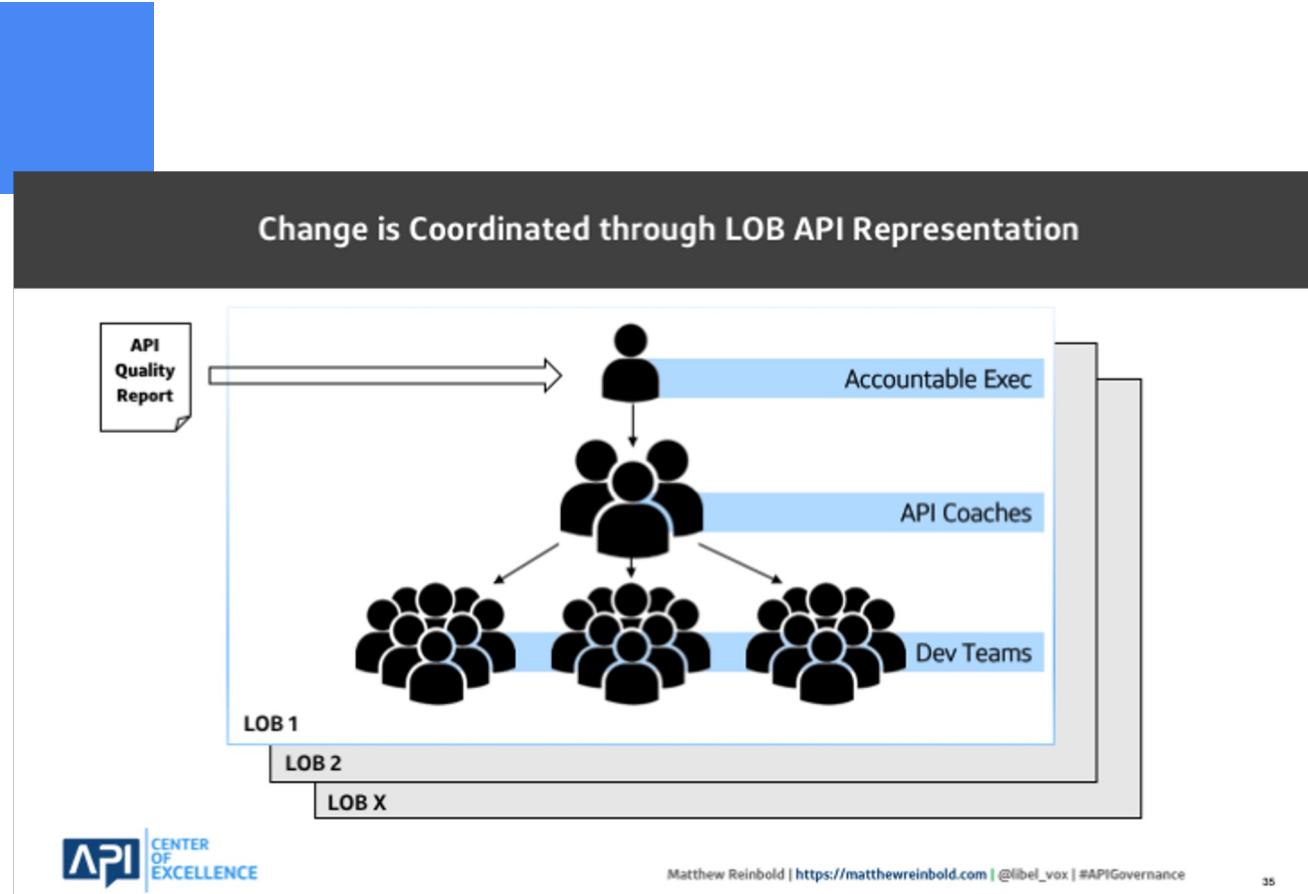
# Gobierno de APIs y servicios

Debemos generar estándares, mejores prácticas y orientación para garantizar el éxito de las APIs en toda la empresa.

La **comunidad de APIs (CoE)** es un equipo de consultoría y servicios formado orgánicamente por equipos participantes.

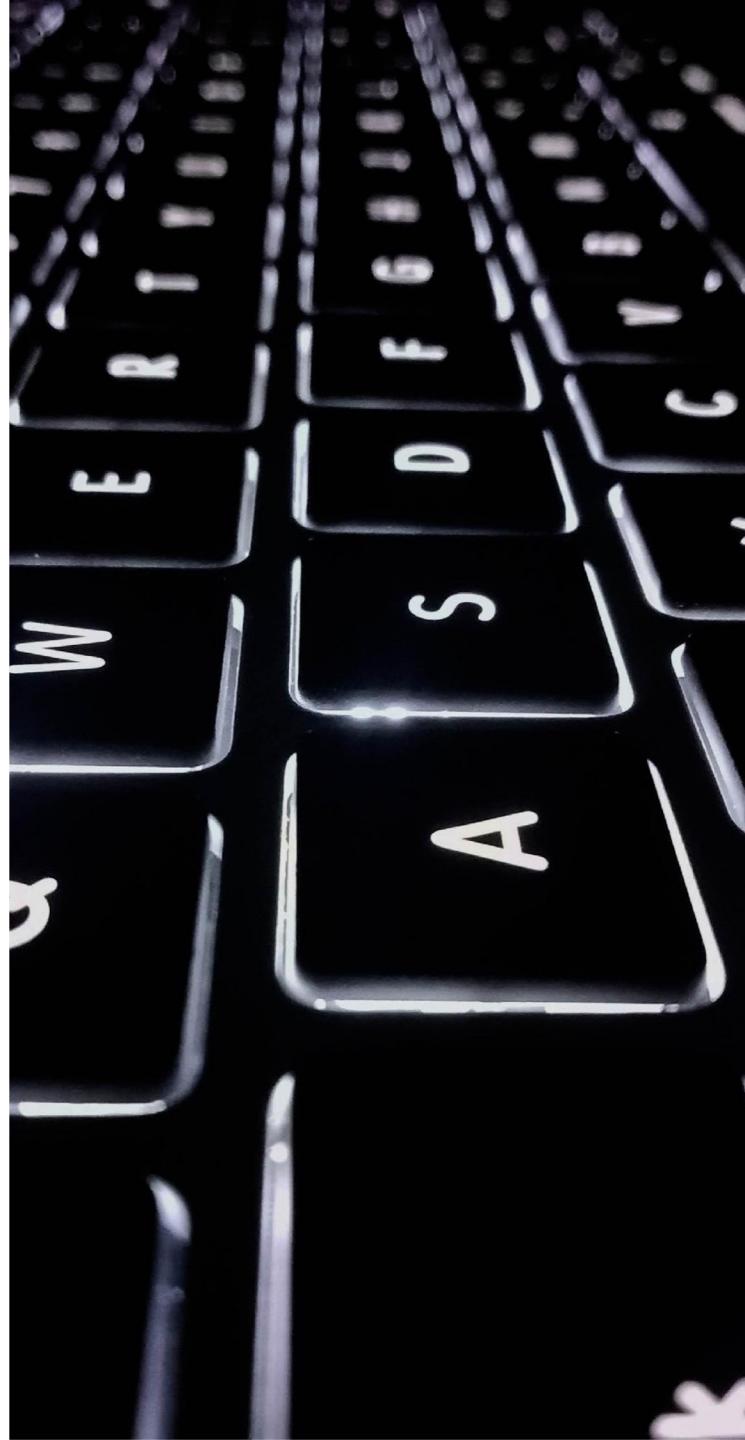
Buscan proporcionar pautas para estándares comunes de reutilización.

El **equipo de excelencia de APIs** posee el ciclo de vida completo de las API, desde el diseño hasta la adopción, y generalmente es bienvenido a lanzar los productos que cree que brindan el mejor valor.



# Módulo I

## API Desing



# Recursos Web / Características



Los recursos son el pilar de sistemas basados en web. La web a menudo es referida como “resource-oriented”.



Un recurso es algo que expones a la Web, desde un documento, un dispositivo, un video o incluso un proceso de negocio.



Desde el punto de vista del consumidor, un recurso es cualquier cosa con la cual se puede interactuar para lograr un objetivo.



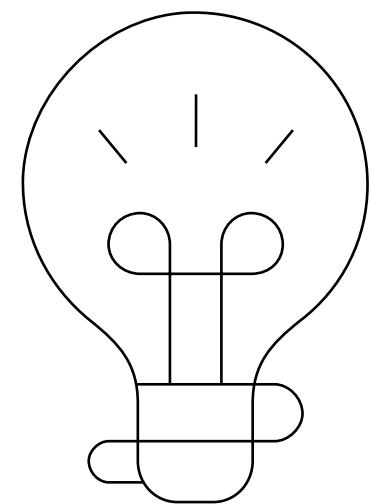
Un recurso debe tener al menos un URI.



Los URI deben ser descriptivos (analizados por humanos) y tener estructura



Los URI no necesitan estructura / con previsibilidad, pero son valiosos (y más fáciles) para que los usuarios naveguen a través de la aplicación



# Recursos Web

## / Nombrado de recursos



/service.do?action=getarticle&id=5  
/data/article/5/4/6/size  
/consultaUsuarios/33245/  
/usuario?rfc=ASDF898989  
/guardaPagos.do?pago=12345  
/realizar\_pagos  
/pago/producto/ahorrapoquito  
/hacer-transaccion/consulta-de-saldos

# **Recursos Web**

## **/ Nombrado de recursos**



[www.bancopoder.com/clientes/v1](http://www.bancopoder.com/clientes/v1)  
[www.bancopoder.com/creditos/v1](http://www.bancopoder.com/creditos/v1)  
[www.bancopoder.com/creditos/v1/aprobaciones](http://www.bancopoder.com/creditos/v1/aprobaciones)  
[www.bancopoder.com/creditos/v1/autorizaciones](http://www.bancopoder.com/creditos/v1/autorizaciones)  
[www.bancopoder.com/creditos/v1/{idCredito}/saldos](http://www.bancopoder.com/creditos/v1/{idCredito}/saldos)  
[www.bancopoder.com/creditos/v1/{idCredito}/pagos](http://www.bancopoder.com/creditos/v1/{idCredito}/pagos)

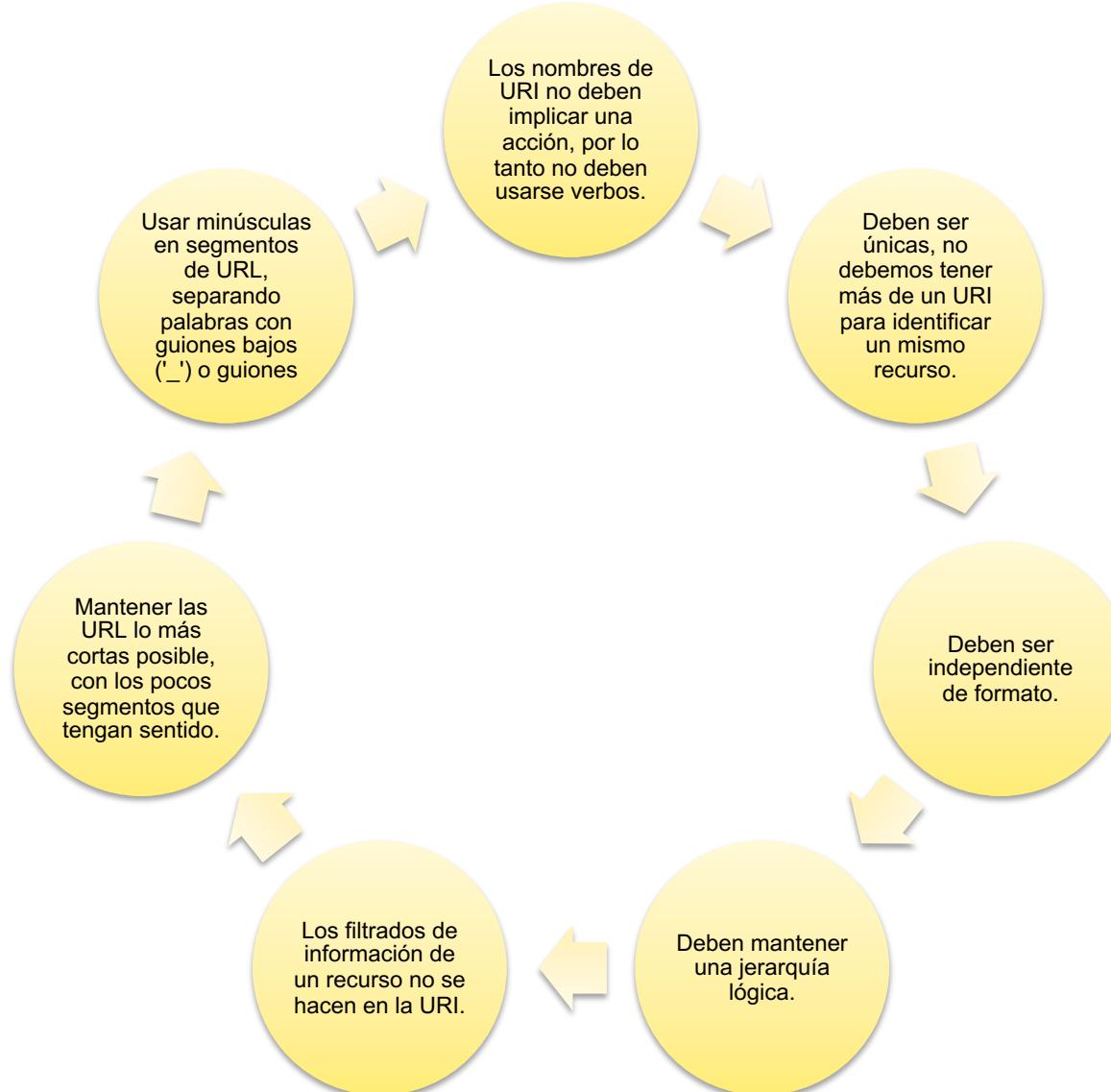
[www.bancopoder.com/inversiones/v1](http://www.bancopoder.com/inversiones/v1)  
[www.bancopoder.com/empleados/v1](http://www.bancopoder.com/empleados/v1)

[www.bancopoder.com usuarios/v1](http://www.bancopoder.com usuarios/v1)  
[www.bancopoder.com usuarios/v1/0001](http://www.bancopoder.com usuarios/v1/0001)

[www.bancopoder.com/pagos/v1](http://www.bancopoder.com/pagos/v1)  
[www.bancopoder.com/pagos/v1/1010](http://www.bancopoder.com/pagos/v1/1010)

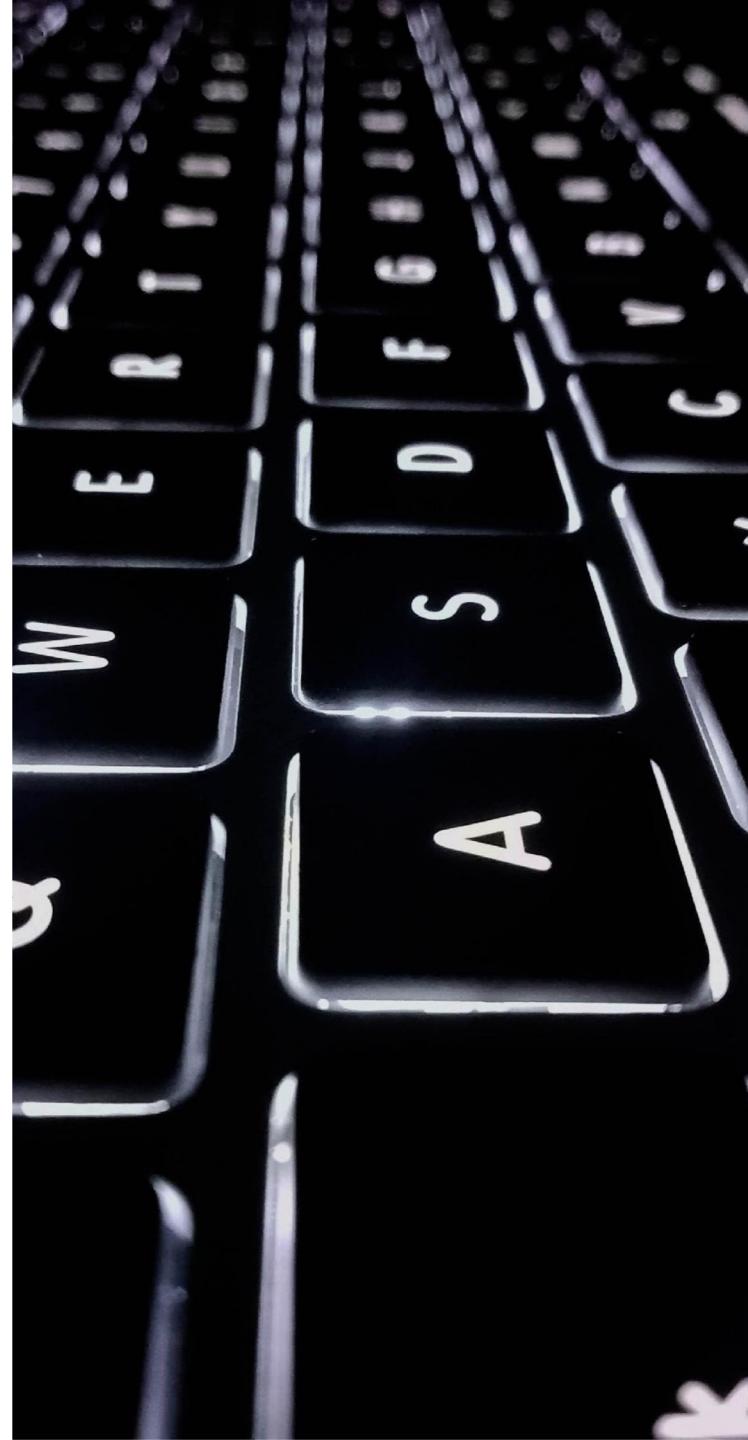
[www.bancopoder.com/productos/v1](http://www.bancopoder.com/productos/v1)  
[www.bancopoder.com/productos/v1/1/creditos](http://www.bancopoder.com/productos/v1/1/creditos)

# Recursos Web / Estandar para la creación



# Módulo I

## REST



# Estilo de Arquitectura

**REST** es un conjunto coordinado de restricciones de arquitectura que intentan minimizar la latencia y comunicación en la red, mientras intentan maximizar la independencia y escalabilidad de la implementación de los componentes.

**REST no es un standard**

**REST no es un protocolo**

API RESTfull aprovecha las características del protocolo de transferencia de hipertexto para aspectos importantes de atender solicitudes y especificar acciones.

Debemos usar identificadores uniformes de recursos (URI) como la dirección de cada recurso único o colección de recursos.

Utilizar los verbos HTTP (GET, POST, PUT, DELETE) para especificar la acción a realizar en un recurso (leer, crear, actualizar, eliminar respectivamente).

Comunicar el estado del recurso utilizando códigos de estado HTTP además de cargas útiles que incluyen descripciones de error más específicas.



**REST API**

# Verbos y Operaciones

Debemos utilizar los códigos de estado HTTP para indicar mediante programación el resultado de una operación.

Según la especificación HTTP, los códigos de estado siempre significan lo mismo, así que siempre debemos asegurarnos de usar el correcto para comunicar lo que la especificación dice que deberían.

Código	Operación	Descripción
<b>200</b>	OK	La solicitud GET se procesó correctamente y la API devuelve el recurso solicitado. Una solicitud PUT se procesó con éxito.
<b>201</b>	CREADO	Se utiliza con una solicitud de creación POST exitosa
<b>204</b>	SIN CONTENIDO	Se utiliza con una solicitud DELETE exitosa.
<b>400</b>	PETICIÓN INCORRECTA	Se utiliza cuando el cliente API intenta una solicitud que no se puede entender. Errores de sintaxis con la solicitud URI que no existe Los valores proporcionados están fuera de rango Faltan datos obligatorios (como en solicitudes POST) Preguntado por algo que viola una regla comercial Parámetros de cadena de consulta en conflicto
<b>401</b>	NO AUTENTICADO	Se utiliza cuando el cliente API no se reconoce, como un token OAuth perdido o caducado.
<b>403</b>	PROHIBIDO	El cliente API no tiene permiso para realizar la acción (por ejemplo, POST o DELETE) El cliente API no tiene permiso para acceder al objeto solicitado
<b>404</b>	NO ENCONTRADO	El recurso solicitado no existe en el servidor. Asume que el URI era válido y normalmente devolvería el objeto solicitado, si existe. No use 404 para URI mal formados (por ejemplo, URI que nunca funcionarían). Use 400 para ese escenario.
<b>429</b>	DEMASIADAS SOLICITUDES	Se usa cuando un cliente API ha excedido sus límites de velocidad de peticiones
<b>500</b>	ERROR INTERNO DE SERVIDOR	Una trampa para todas las cosas que salen mal en el servidor.

# Operaciones CRUD

Verbo	Acción
GET	<p>Recuperar un recurso del servidor El uso repetido siempre devuelve el mismo objeto (por ID; las propiedades específicas pueden cambiar).</p> <p>Esta acción nunca cambia el objeto que se solicita.</p>
POST	<p>Crear un nuevo recurso en el servidor El uso repetido con la mismo payload creará un objeto idéntico con un nuevo ID.</p> <p>El payload de respuesta debe incluir una copia del recurso creado (con el ID).</p>
PUT	<p>Almacene una nueva versión completa de un recurso existente en el servidor El payload es el objeto completo en su nuevo formato.</p> <p>El uso repetido simplemente sobrescribe el objeto con el payload, no se crea nada nuevo.</p>
DELETE	<p>Eliminar un recurso del servidor.</p> <p>Dos opciones aceptadas para uso repetido: Si el objeto realmente existe pero está marcado como eliminado, esto no tiene efecto</p> <p>Si el objeto se elimina realmente del sistema, esto puede devolver un 404 en solicitudes posteriores.</p>

# Ejercicio práctico

## Diseñar paths

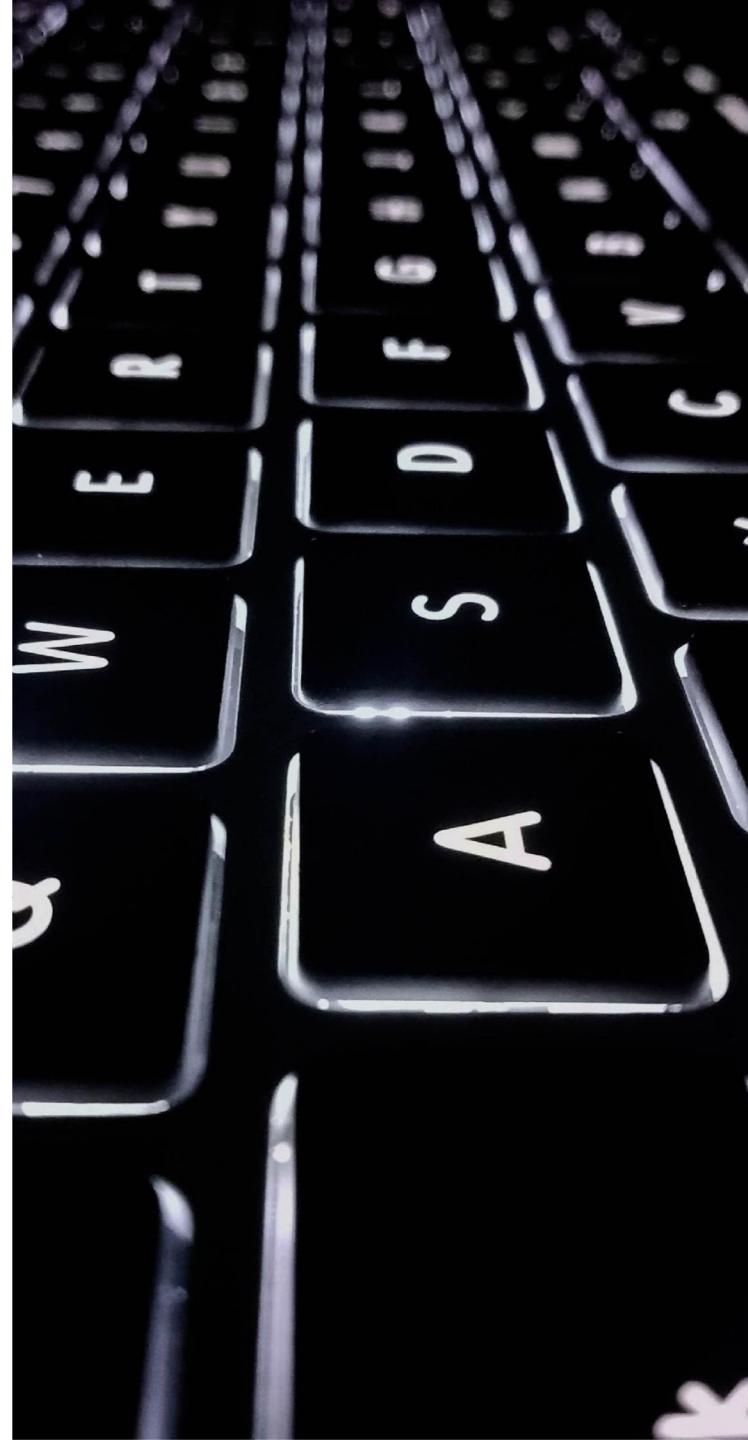
```
userDetailsCardOnHover = showOnHover(UserDetailsCard);  
  
UserLink = ({  
  //  
  secondaryLink,  
  children,  
  userAvatar,  
  type,  
  ...  
  style: {  
    ...  
  }  
})  
  
includeAvatar && (  
  <UserDetailsCardOnHover  
    user={user}  
    delay={CARD_HOVER_DELAY}  
    wrapperClassName={styles.avatarContainer}  
  >  
  <Avatar user={user} />  
)</UserDetailsCardOnHover>  
  
div  
  className={classNames(  
    styles.linkContainer,  
    inline && styles.inlineContainer  
  )}  
  
<UserDetailsCardOnHover user={user} delay={CARD_HOVER_DELAY}>  
  <Link  
    to={({ pathname: buildUserUrl(user) })}  
    className={classNames(styles.name, {  
      [styles.alt]: type === 'alt',  
      [styles.secondaryName]: !secondaryLink,  
      [styles.inlineLink]: inline  
    })}  
  >  
    {children}  
  </Link>  
  
  {!secondaryLink  
   ? null  
   : <a  
     href={secondaryLink.href}  
     className={classNames(styles.name, {  
       [styles.alt]: type === 'alt',  
       [styles.secondaryLink]: secondaryLink,  
     })}  
   >  
    {secondaryLink.label}  
  </a>  
  }  
</UserDetailsCardOnHover>  
</div>  
span>
```

```
145      </a>  
146    </li>  
147  </ul>  
148 </div>  
149 );  
150 }  
151  
152 <renderWhatsNewLinks() =>  
153  return () =>  
154    <div className={styles.whatsNewList}>  
155      <h4 className={styles.whatsNewSection}>  
156        <span>What's new</span>  
157      </h4>  
158      <ul className={styles.whatsNewListItems}>  
159        {this.renderWhatsNewItem(  
160          {this.renderWhatsNewItem(  
161            {this.renderWhatsNewItem(  
162              {this.renderWhatsNewItem(  
163                {this.renderWhatsNewItem(  
164                  {this.renderWhatsNewItem(  
165                    {this.renderWhatsNewItem(  
166                      {this.renderWhatsNewItem(  
167                        {this.renderWhatsNewItem(  
168                          {this.renderWhatsNewItem(  
169                            {this.renderWhatsNewItem(  
170                            <li className={styles.footerList}>  
171                              <a href={trackUrl(url)}  
172                                target="_blank"  
173                                rel="noopener noreferrer"  
174                              >  
175                                {title}  
176                              </a>  
177                            </li>  
178                          );  
179                        );  
180                      );  
181                    );  
182                  );  
183                );  
184              );  
185            );  
186          );  
187        );  
188      );  
189    );  
190  );  
191  );  
192  );  
193  );  
194  );  
195  );  
196  );  
197  );  
198 <render() =>  
199  return () =>  
200    <footer className={styles.footerGlobal}>  
201      <div className="container">  
202        {this.renderFooterMain()}  
203        {this.renderFooterSub()}  
204      </div>  
205    </footer>  
206  );
```

# Módulo I

## OpenAPI 2.0, 3.0

### Swagger

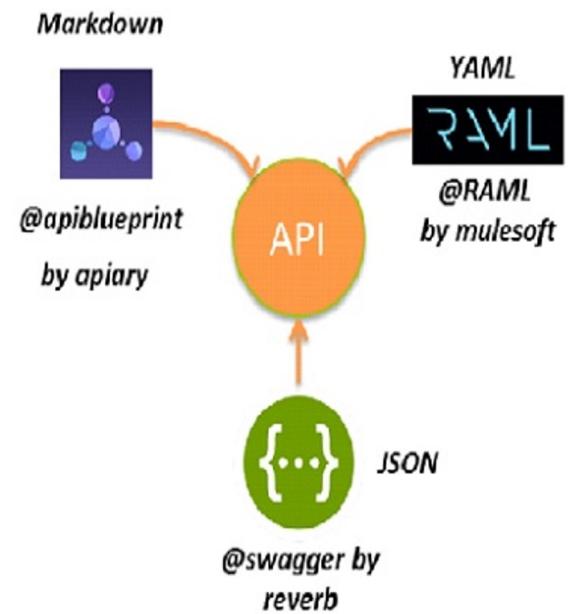


# OpenAPI OAS

La especificación OpenAPI (OAS) define una descripción de interfaz estándar, independiente del lenguaje de programación para las APIs REST.

Permite a los humanos y las computadoras descubrir y comprender las capacidades de un servicio sin requerir acceso al código fuente, documentación adicional o inspección del tráfico de red.

Cuando se define correctamente a través de OpenAPI, un consumidor puede comprender e interactuar con el servicio remoto con una cantidad mínima de lógica de implementación



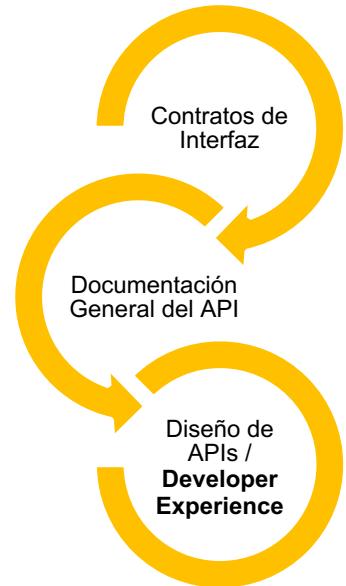
# Swagger

Cuando hablamos de Swagger nos referimos a una **serie de reglas, especificaciones y herramientas** que nos ayudan a documentar nuestras APIs. YAML

De esta manera, podemos realizar documentación que sea realmente útil para las personas que la necesitan. Swagger nos ayuda a crear documentación que todo el mundo entienda.

Formatos:

- YAML o JSON



## Descripción de APIs

## Información, metadata

## Versión

## Esquema de Seguridad

## basePath

## Host

## Version de Spect

## Esquemas de protocolo

```
! clientes-v1.yaml •
1  # Example YAML to get you started quickly.
2  # Be aware that YAML has indentation based coping.
3  # Code completion support is available so start typing for available options.
4  swagger: "2.0"
5
6  # This is your document metadata
7  info:
8    version: "0.0.1"
9    title: API de Clientes del banco PODER
10   description: |
11     API para poder realizar la captacion de nuevos clientes del banco poder
12
13  host:
14    www.bancopoder.com
15  schemes:
16    - https
17
18  basePath: /clientes/v1
19
20  securityDefinitions:
21    Bearer:
22      description: ''
23      Para acceder a la API se debe pasar un token válido en cada petición.
24      Se debe utilizar en el encabezado 'Authorization' :
25      | Bearer: 2xgsLJmBAtTNIU8ngge8XbmDvaNGsx
26      type: apiKey
27      name: Authorization
28      in: header
29
30  security:
31    - Bearer: []
```

# APIs como contratos de Interfaz

Cuando se genere un API, se debe considerar como un contrato para sus desarrolladores consumidores (internos y externos) y evitar cambios importantes.

Los clientes gastan mucho tiempo y dinero en integrarse con una API, y esperan que el API permanezca igual mientras sea compatible.

Los cambios en una API que “rompan” la aplicación de un cliente pueden tener un impacto significativo en los mapas y la reputación de los productos del cliente.

## **Diseño de las APIs como un contrato**

En el contrato	Fuera del contrato
Todos los aspectos de la URL	El orden de las propiedades
Nombres de propiedades	Propiedades y parámetros de consulta aún no definidos (se pueden agregar nuevos a una API existente)
La existencia de las propiedades definidas y los parámetros de consulta	El orden de clasificación de varios objetos en una sola respuesta
La disponibilidad de un formato de respuesta particular para un punto final particular	El número de objetos por respuesta paginada
Códigos de estado HTTP (incluyendo códigos de error)	Formatos de hipervínculos a otros recursos
La existencia de encabezados HTTP directamente relacionados con el procesamiento de los resultados (ubicación, rango)	Encabezados HTTP relacionados con la transacción (por ejemplo, CORS, Gzip)
	Zonas horarias

# Ejemplos de Swagger 2.0 y 3.0

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training\\_apis\\_apigee\\_enero\\_2024/01\\_openapi/lab1/openapi-overview](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training_apis_apigee_enero_2024/01_openapi/lab1/openapi-overview)

1.- petstore-minimal2.0-v1.yaml

```
145      </a>
146    </li>
147  </ul>
148 </div>
149  );
150}
151
152  renderWhatsNewLinks() {
153    return (
154      <div className={styles.whatsNew}>
155        <h4 className={style}>
156          What's new
157        </h4>
158        {this.renderWhat(<h3>What's new</h3>)}
159        {this.renderWhat(<h3>What's new</h3>)}
160        {this.renderWhat(<h3>What's new</h3>)}
161        {this.renderWhat(<h3>What's new</h3>)}
162      </div>
163    );
164  }
165
166  renderWhatsNewItem(title, url) {
167    return (
168      <li className={styles.footerList}>
169        <a href={trackUrl(url)} target="_blank" rel="noopener noreferrer">
170          {title}
171        </a>
172      </li>
173    );
174  }
175
176  renderFooterSub() {
177    return (
178      <div className={styles.footerSub}>
179        <Link to="/" title="Home - Unsplash">
180          <Icon type="logo" alt="Unsplash logo" />
181        </Link>
182        <span className={styles.footerSlogan}>
183          A community for creative people
184        </span>
185      </div>
186    );
187  }
188
189  render() {
190    return (
191      <footer className={styles.footerGlobal}>
192        <div className="container">
193          {this.renderFooterMain()}
194          {this.renderFooterSub()}
195        </div>
196      </footer>
197    );
198  }
199}
200
201  renderFooterMain() {
202    return (
203      <div>
204        {this.renderFooterLinks()}
205      </div>
206    );
207  }
208
209  renderFooterLinks() {
210    return (
211      <ul>
212        {this.renderFooterLink(<li>About</li>)}
```

# Ejemplos de Swagger 2.0 y 3.0

- 1.- petstore-minimal2.0-v1.yaml
- 2.- petstore-expanded2.0-v1.yaml
- 3.- armar-tu-especificacion.yaml
- 4.- petstore-simple2.0-v1.yaml
- 5.- sample2.0-v1.yaml
- 6.- uber2.0-v1.yaml

## Spec 3

- 1.- basic-structure3.0-v1.yaml
- 2.- cambios3.0-v1.yaml
- 3.- armar-tu-especificacion.yaml
- 4.- petstore-v1.yaml
- 5.- sample3.0-v1.yaml

Interfaz de clientes:  
clientes-v1.yaml

```
145      </a>
146    </li>
147  </ul>
148 </div>
149  );
150}
151
152  renderWhatsNewLinks() {
153    return (
154      <div className={styles.container}>
155        <h4>What's new</h4>
156        <ul className={styles.list}>
157          {this.renderWhat(
158            this.renderWhat(
159              this.renderWhat(
160                this.renderWhat(
161                  this.renderWhat(
162                    this.renderWhat(
163                      this.renderWhat(
164                        this.renderWhat(
165                          <li>
166                            <a href="#">{title}</a>
167                          </li>
168                        )>
169                      </ul>
170                    )>
171                    <div>
172                      <h4>New item</h4>
173                      <ul>
174                        <li>
175                          <a href="#">{title}</a>
176                        </li>
177                      </ul>
178                    </div>
179                  )>
180                  <div>
181                    <h4>Footer</h4>
182                    <ul>
183                      {this.renderFooterSub()}
184                    </ul>
185                  </div>
186                )>
187                <div>
188                  <Link to="/" title="Home - Unsplash">
189                    <Icon type="logo" className={styles.footerSubLogo}>
190                    </Icon>
191                  </Link>
192                  <span className={styles.footerSlogan}>A free image library for everyone</span>
193                </div>
194              )>
195            )>
196          )>
197        )>
198      </div>
199    )>
200  </div>
201  <div>
202    {this.renderFooterMain()}
203    {this.renderFooterSub()}
204  </div>
205 </div>
206 );
```

# Hands-ON Diseño de SPECS OPENAPI 2.0 y 3.0

<https://swagger.io/docs/specification/2-0/basic-structure/>

# Hands-ON Startup de Crédito online - Banco PODER

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training\\_apigee\\_microservicios\\_enero\\_2022](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training_apigee_microservicios_enero_2022)

```
userDetailsCardOnHover = showOnHover(UserDetailsCard);

UserLink = {
  ...
  secondaryLink,
  children,
  userAvatar,
  ...
}

includeAvatar && (
  userDetailsCardOnHover
  user={user}
  delay={CARD_HOVER_DELAY}
  wrapper={UserDetailsCardWrapper}
)
  <Avatar user={user} />
</UserDetailsCardOnHover>
)



<UserDetailsCardOnHover user={user} delay={CARD_HOVER_DELAY}>
    <Link
      to={({ primaryName }) => buildUserUrl(user)}
      className={classnames(styles.linkLabel, [styles.altType, type], [styles.inlineLink, inline])}
    >
      {children || user.name}
    </Link>
  <!secondaryLink
    ? null
    : <a
        href={secondaryLink.href}
        className={classnames(styles.name, [styles.secondaryLinkType, type], [styles.secondaryLinkColor, color])}
      >
        {secondaryLink.label}
      </a>
    </UserDetailsCardOnHover>
  </div>
  span>


```

```
145   </a>
146   </li>
147   </ul>
148   </div>
149 }
150 }
151
152 renderWhatsNewLinks() {
153   return (
154     <div className={styles.whatsNewItemList}>
155       <h4 className={styles.whatsNewItemSection}>
156         <ul className={classNames(styles.whatsNewItemList, [this.renderWhatSectionColor])}>
157           {this.renderWhatsNewLinksItem()}
158           {this.renderWhatsNewLinksItem()}
159           {this.renderWhatsNewLinksItem()}
160           {this.renderWhatsNewLinksItem()}
161           {this.renderWhatsNewLinksItem()}
162           {this.renderWhatsNewLinksItem()}
163           {this.renderWhatsNewLinksItem()}
164         </ul>
165       </div>
166     );
167   }
168 }
169
170 renderWhatsNewItem(title, url) {
171   return (
172     <li className={styles.whatsNewItem}>
173       <a href={url}>
174         <span>{title}</span>
175         <a href="#" target="_blank" rel="noopener noreferrer">
176           <img alt="link icon" />
177         </a>
178       </a>
179     </li>
180   );
181 }
182
183
184 renderFooterSub() {
185   return (
186     <div className={styles.footerSub}>
187       <Link to="/" title="Home - Unsplash">
188         <Icon
189           type="logo"
190           className={styles.footerSubLogo}
191         />
192       </Link>
193       <span className={styles.footerSlogan}>
194         <img alt="Unsplash logo" />
195       </span>
196     </div>
197   );
198 }
199
200 render() {
201   <div id="global">
202     <div className="container">
203       {this.renderFooterMain()}
204       {this.renderFooterSub()}
205     </div>
206   </div>
207 }


```

# Laboratorio I

Simularemos diseñar un sistema basado en APIs, el cual genere el funcionamiento de un **Banco**, bajo la estrategia de **Contract First** y **API First Development**.

En este caso: **Banco Poder**, el cual tendrá diferentes líneas de negocio o productos a ofrecer a clientes.

Diseño de especificaciones OPENAPI para poder crear diferentes productos que ofrecerá **Banco Poder**.

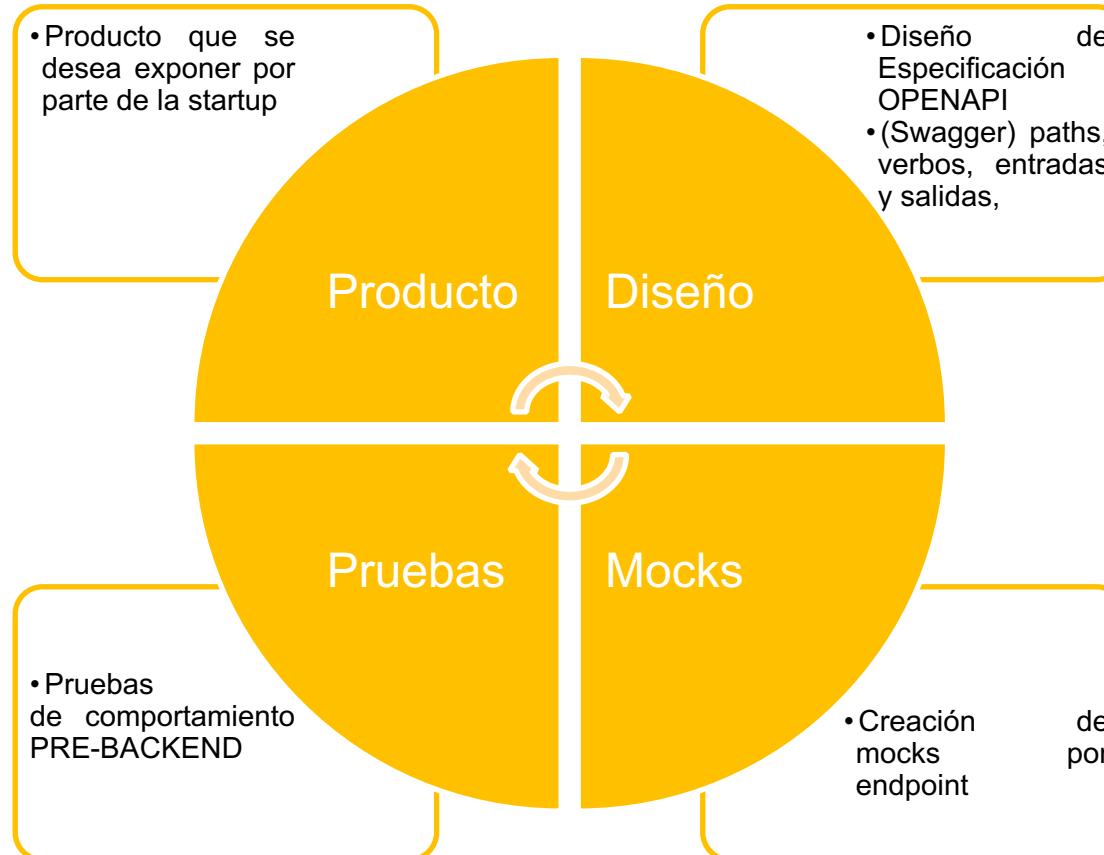
APIs a diseñar:

- Empleados
  - API con el objetivo de poder dar de **alta los empleados** del Banco Poder, así como **diferentes operaciones** necesarias usadas dentro del mismo.
- Clientes
  - API con el objetivo de poder generar clientes del banco para ofrecer diferentes productos como de **créditos, inversiones, afore etc.**
- Créditos
  - API con el objetivo de poder ofrecer **diferentes productos** de crédito a los clientes del banco.
- Seguros Créditos
  - API con el objetivo de generar un seguro para poder salvaguarda el incumplimiento del crédito

# Laboratorio I

**APIs** con el objetivo de poder ofrecer el servicio de producto de **crédito para clientes de Banco Poder**, así como la administración de empleados dentro del banco.

API Empleados  
API Clientes  
API Créditos  
API Seguros



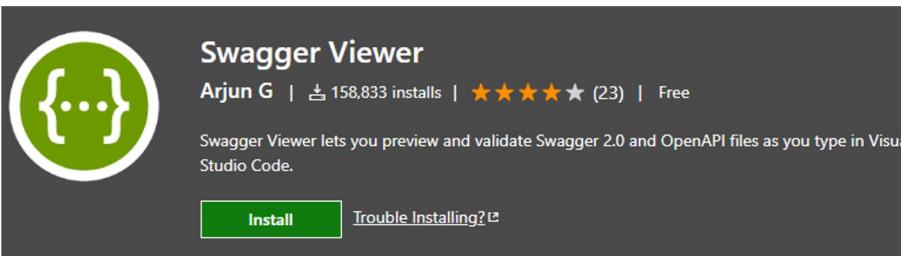
# Herramientas de desarrollo



apimocker build passing

**npm** npm install apimocker -g  
10 dependencies 2 dependents version 1.1.1 updated 3 months ago

6 ★

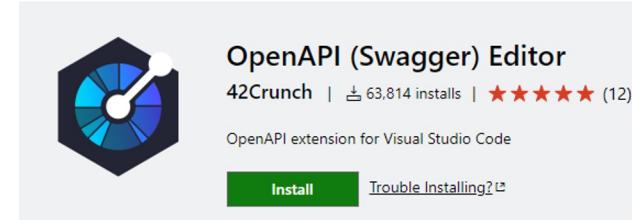


**Swagger Viewer**  
Arjun G | 158,833 installs | ★★★★★ (23) | Free

Swagger Viewer lets you preview and validate Swagger 2.0 and OpenAPI files as you type in Visual Studio Code.

[Install](#) [Trouble Installing?](#)

cucumber



**OpenAPI (Swagger) Editor**  
42Crunch | 63,814 installs | ★★★★★ (12) |

OpenAPI extension for Visual Studio Code

[Install](#) [Trouble Installing?](#)



# Hands-ON SPEC de Clientes

<https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training/apis/apigee/enero/2024/lab1>

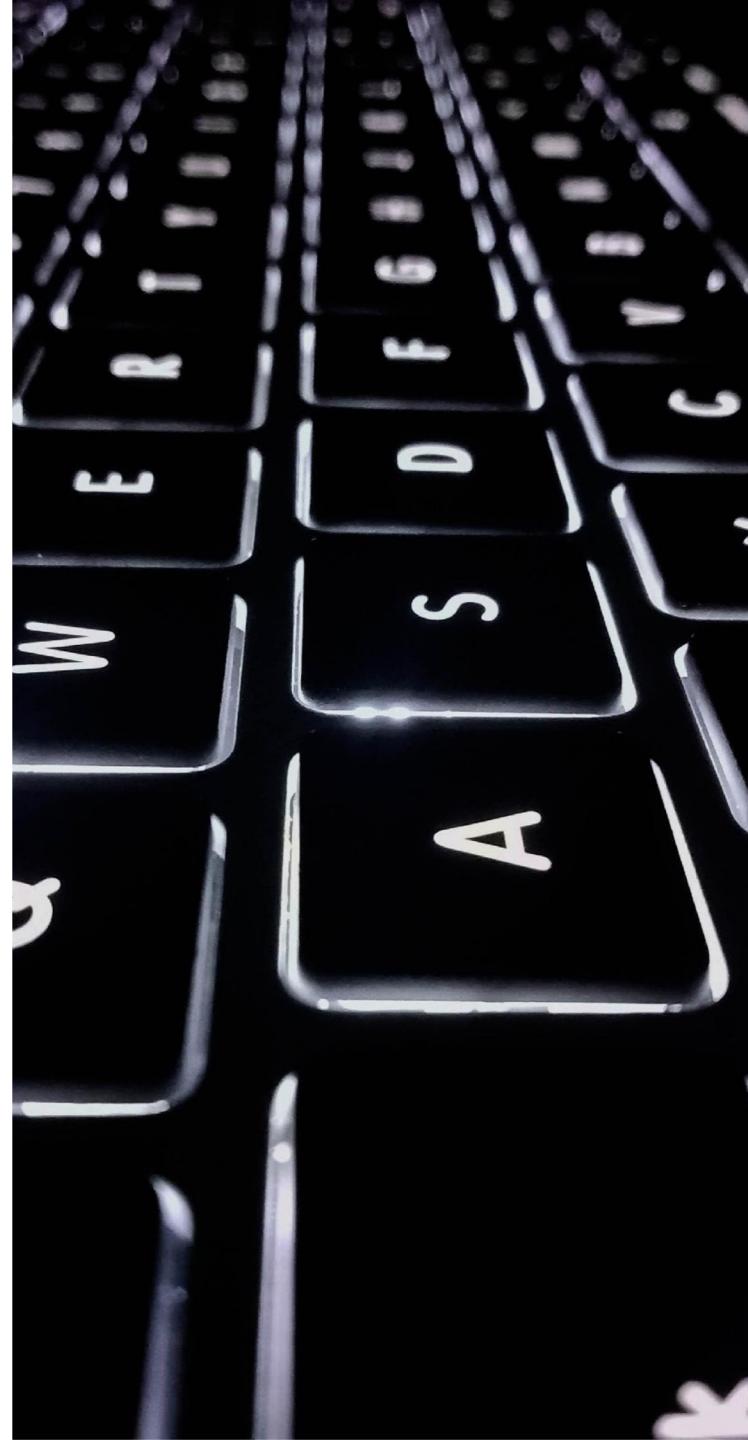
```
userDetailsCardOnHover = showOnHover(UserDetailsCard);

UserLink = ({

  secondaryLink,
  children,
  userAvatar,
  name,
  ...rest
}) => {
  return (
    <div className={styles.container}>
      {includeAvatar && (
        <UserDetailsCardOnHover
          user={user}
          delay={CARD_HOVER_DELAY}
          wrapperClassName={styles.avatarContainer}
        >
          <Avatar user={user} />
        </UserDetailsCardOnHover>
      )}
      <div
        className={classNames(
          styles.linkContainer,
          inline && styles.inlineLink
        )}
      >
        <UserDetailsCardOnHover user={user} delay={CARD_HOVER_DELAY}>
          <Link
            to={{ pathname: buildUserUrl(user) }}
            className={classNames(styles.name, {
              [styles.alt]: type === 'alt',
              [styles.centerName]: !secondaryLink,
              [styles.inlineLink]: inline,
            })}
          >
            {children || user.name}
          </Link>
        </UserDetailsCardOnHover>
        {!secondaryLink ? null : (
          <a href={secondaryLink.href} className={classNames([
            styles.secondaryLink,
            [secondaryLink.type] === 'link' ? styles.link : styles.secondaryLink
          ])}>
            {secondaryLink.label}
          </a>
        )}
      </UserDetailsCardOnHover>
    </div>
    <span>
      {145}   </a>
      {146}   </li>
      {147} </ul>
      {148} </div>
      {149} );
      {150} }
      {151} renderWhatsNewLinks() {
      {152}   return (
      {153}     <div className={styles.whatsNew}>
      {154}       <h4 className={style}>
      {155}         What's new
      {156}       </h4>
      {157}       {this.renderWhat(<div>)}
      {158}       {this.renderWhat(<div>)}
      {159}       {this.renderWhat(<div>)}
      {160}       {this.renderWhat(<div>)}
      {161}       {this.renderWhat(<div>)}
      {162}       {this.renderWhat(<div>)}
      {163}       {this.renderWhat(<div>)}
      {164}       {this.renderWhat(<div>)}
      {165}     </ul>
      {166}   </div>
      {167} );
      {168} }
      {169} renderWhatsNewItem(title, url) {
      {170}   return (
      {171}     <li className={styles.footerItem}>
      {172}       <a href={trackUrl(url)} target="_blank" rel="noopener noreferrer">
      {173}         {title}
      {174}       </a>
      {175}     </li>
      {176} );
      {177} }
      {178} renderFooterSub() {
      {179}   return (
      {180}     <div className={styles.footerSub}>
      {181}       <Link to="/" title="Home - Unsplash">
      {182}         <Icon type="logo" className={styles.footerSubLogo}>
      {183}           </Icon>
      {184}         </Link>
      {185}       <span className={styles.footerSlogan}>
      {186}         <img alt="Unsplash logo" />
      {187}         Welcome to Unsplash!
      {188}       </span>
      {189}     </div>
      {190}   );
      {191} }
      {192} render() {
      {193}   return (
      {194}     <footer className={styles.footerGlobal}>
      {195}       <div className="container">
      {196}         {this.renderFooterMain()}
      {197}         {this.renderFooterSub()}
      {198}       </div>
      {199}     </footer>
      {200}   );
      {201} }
```

# Módulo I

## Mocks



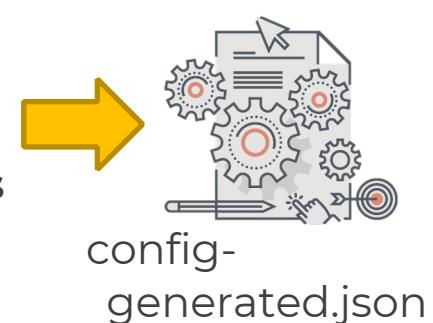
# Estrategia de Mocking

Es un objeto preprogramado que simula el **comportamiento de un objeto** abstracto de la vida real.



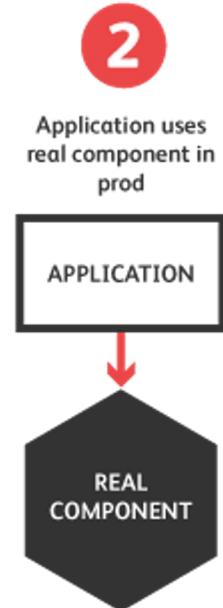
## npm APIMocker

Es un módulo node.js que ejecuta un servidor http, el cual **simula respuestas de servicios o APIs.**



Por cada endpoint/recursos:

1. Verbos HTTP
2. Parametro/Payload
3. Parametro: valores(código http y respuesta mock).



Por cada parametro:valor:

1. Respuesta Mock JSON.

{JSON}

# Hands-ON API de Clientes Mocks

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training\\_apis\\_apigee\\_enero\\_2024/01/openapi/lab/2/api clientes mocks](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training_apis_apigee_enero_2024/01/openapi/lab/2/api%20clientes%20mocks)

```
145      </a>
146    </li>
147  </ul>
148 </div>
149  );
150}
151
152  renderWhatsNewLinks() {
153    return (
154      <div className={styles.whatsNew}>
155        <h4 className={styles.h4}>What's new</h4>
156        <ul className={styles.list}>
157          {this.renderWhat(<li>)}
158          {this.renderWhat(<li>)}
159          {this.renderWhat(<li>)}
160          {this.renderWhat(<li>)}
161          {this.renderWhat(<li>)}
162          {this.renderWhat(<li>)}
163          {this.renderWhat(<li>)}
164        </ul>
165      </div>
166    );
167  }
168
169  renderWhatsNewItem(title, url) {
170    return (
171      <li className={styles.footerItem}>
172        <a href={url} target="_blank" rel="noopener noreferrer">
173          {title}
174        </a>
175      </li>
176    );
177  }
178
179  renderFooterSub() {
180    return (
181      <div className={styles.footerSub}>
182        <Link to="/" title="Home - Unsplash">
183          <Icon type="logo" className={styles.footerSubLogo}>
184        </Link>
185        <span className={styles.footerSlogan}>
186          "A free image library for everyone"
187        </span>
188      </div>
189    );
190  }
191
192  render() {
193    return (
194      <footer className={styles.footerGlobal}>
195        <div className="container">
196          {this.renderFooterMain()}
197          {this.renderFooterSub()}
198        </div>
199      </footer>
200    );
201  }
202}
203
204
205
206  );
```

# Hands-ON API de Clientes Mocks

npm run init-api

<https://github.com/gstrup/apimocker>

<https://github.com/kent-wu/gulp-apimocker>

<https://www.urlencoder.org/>

probar : localhost:8000/first

```
userDetailsCardOnHover = showOnHover(UserDetailsCard);

UserLink = {
  ...
  secondaryLink,
  children,
  userAvatar,
  ...
  (
    ...
  )
}

includeAvatar && (
  userDetailsCardOnHover
  user={user}
  delay={CARD_HOVER_DELAY}
  wrapperClassName={styles.avatarContainer}
)
  >
  <Avatar user={user} />
</UserDetailsCard>



<UserDetail>
    <Link
      to={{ pathname: buildUserUrl(user) }}
      className={classNames(styles.name, {
        [styles.alt]: type === 'alt',
        [styles.secondaryLink]: secondaryLink,
      })}
      ...
    >
      {children || user.name}
    </Link>
  </UserDetail>
  {!secondaryLink
    ? null
    : <a
        href={secondaryLink.href}
        className={classNames(styles.name, {
          [styles.alt]: type === 'alt',
          [styles.secondaryLink]: secondaryLink,
        })}
        ...
      >
        {secondaryLink.label}
      </a>
  }
</UserDetailsCardOnHover>
</div>


```

```
145       </a>
146     </li>
147   </ul>
148 </div>
149   );
150 }
151
152 <renderWhatsNewLinks() >
153   return (
154     <div className={styles.whatsNew}>
155       <h4 className={style}>
156         <ul className={classNames(
157           styles.whatsNewList,
158           <this.renderWhatsNewList() />
159         )}>
160           <this.renderWhatsNewItem() />
161           <this.renderWhatsNewItem() />
162           <this.renderWhatsNewItem() />
163           <this.renderWhatsNewItem() />
164           <this.renderWhatsNewItem() />
165         </ul>
166       </div>
167     );
168   }
169
170 <renderWhatsNewItem(title, url) >
171   return (
172     <li className={styles.footerItem}>
173       <a href={trackUrl(url)} target="_blank" rel="noopener noreferrer">
174         {title}
175       </a>
176     </li>
177   );
178
179 <renderFooterSub() >
180   return (
181     <div className={styles.footerSub}>
182       <Link to="/" title="Home - Unsplash">
183         <Icon
184           type="logo"
185           className={styles.footerSubLogo}
186         />
187       </Link>
188       <span className={styles.footerSlogan}>
189         <Icon
190           type="heart"
191           className={styles.footerSubIcon}
192         />
193         <span>A free image library for everyone</span>
194       </span>
195     </div>
196   );
197
198 <render() >
199   return (
200     <footer className={styles.footerGlobal}>
201       <div className="container">
202         <this.renderFooterMain() />
203         <this.renderFooterSub() />
204       </div>
205     </footer>
206   );
207
```

# Hands-ON

# API de Empleados

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training\\_apis\\_apigee](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training_apis_apigee) enero 2024/01 openapi/lab3/api empleados mocks

# Hands-ON API de Créditos

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training\\_apis\\_apigee\\_enero\\_202401\\_openapi/lab4](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training_apis_apigee_enero_202401_openapi/lab4)

# Hands-ON APIs sucursales, cuentas, tarjetas

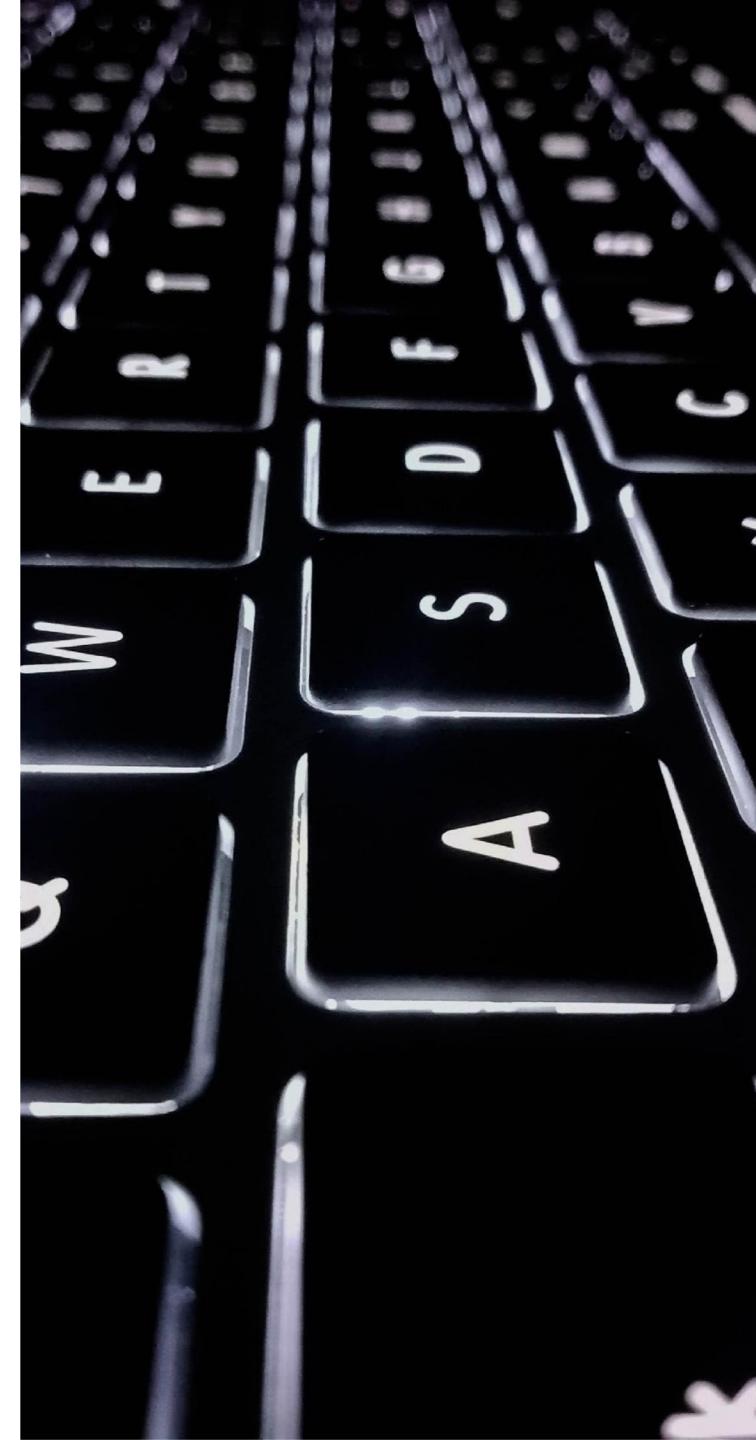
[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training\\_apis\\_apigee](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training_apis_apigee) enero 2024/01 openapi/lab4 ejercicio

```
145      </a>
146    </li>
147  </ul>
148 </div>
149  );
150}
151
152  renderWhatsNewLinks() {
153    return (
154      <div className={styles.whatsNew}>
155        <h4 className={styles.h4}>Lo más reciente</h4>
156        <ul className={styles.list}>
157          {this.renderWhat(<Link to="/>)}
158          {this.renderWhat(<Link to="/>)}
159          {this.renderWhat(<Link to="/>)}
160          {this.renderWhat(<Link to="/>)}
161          {this.renderWhat(<Link to="/>)}
162          {this.renderWhat(<Link to="/>)}
163          {this.renderWhat(<Link to="/>)}
164        </ul>
165      </div>
166    );
167  }
168
169  renderWhatsNewItem(title, url) {
170    return (
171      <li className={styles.listItem}>
172        <a href={url} rel="noopener noreferrer" title="Ir a la página de {title}">
173          {title}
174        </a>
175      </li>
176    );
177  }
178
179  renderFooterSub() {
180    return (
181      <div className={styles.footerSub}>
182        <Link to="/" title="Home - Unsplash">
183          <Icon type="logo" className={styles.footerSubLogo}>
184        </Icon>
185        <span className={styles.footerSlogan}>Unsplash</span>
186      </div>
187    );
188  }
189
190  render() {
191    return (
192      <footer className={styles.footerGlobal}>
193        <div className="container">
194          {this.renderFooterMain()}
195          {this.renderFooterSub()}
196        </div>
197      </footer>
198    );
199  }
200}
```

# Módulo I

## BDD – Behavior Driven Development

Gherkin  
Cucumber

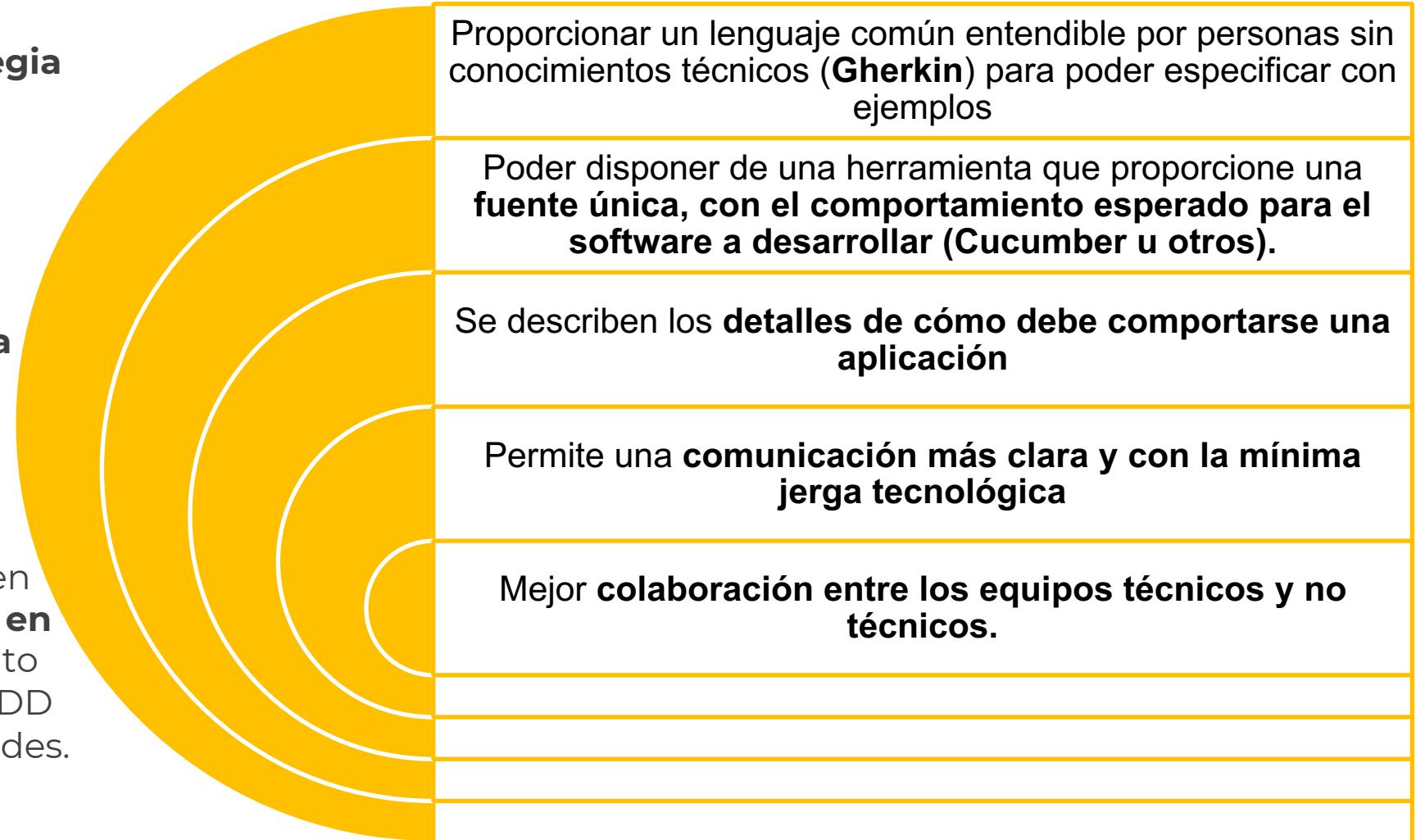


# Desarrollo guiado por comportamiento

BDD (Behavior Driven Development), es una **estrategia de desarrollo** dirigido por comportamiento.

BDD se define en un idioma común entre todos los stakeholders, lo que **mejora la comunicación** entre equipos tecnológicos y no técnicos.

Las pruebas se deben definir antes del desarrollo, aunque en BDD, **las pruebas se centran en el usuario** y el comportamiento del sistema, a diferencia del TDD que se centra en funcionalidades.



# Desarrollo guiado por comportamiento

## Ventajas

Ya no estás definiendo “pruebas”, sino “comportamientos”.

Mejora la comunicación entre desarrolladores, testers, usuarios y la dirección.

Debido a que BDD se especifica utilizando un lenguaje simplificado y común, la curva de aprendizaje es mucho más corta que TDD.

Como su naturaleza no es técnica, puede llegar a un público más amplio.

El enfoque de definición ayuda a una aceptación común de las funcionalidades previamente al desarrollo.

Esta estrategia encaja bien en las metodologías ágiles, ya que en ellas se especifican los requisitos como historias de usuario y de aceptación.

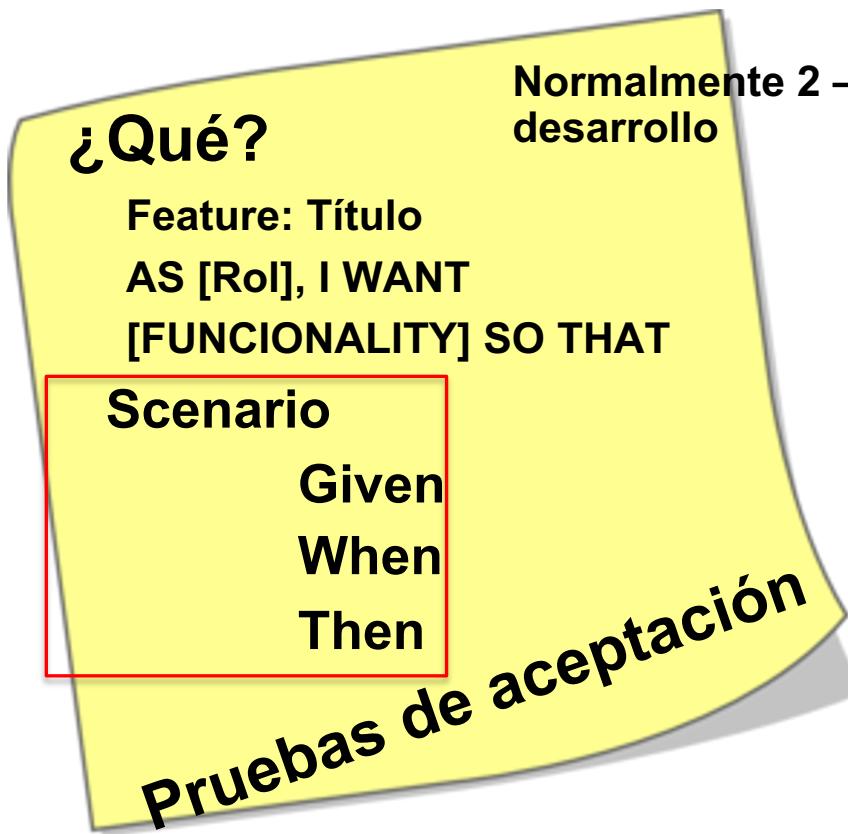
BDD te permite desarrollar, probar y pensar el código desde la perspectiva del usuario.

## ¿Cómo funciona BDD?

Se describe funcionalmente el sistema utilizando “Features” que contienen “Scenarios” o ejemplos concretos de cómo se comportará dicha funcionalidad una vez implementada. Los Scenarios a su vez, están formados por “Steps” (siguiendo la estructura Given, When y Then) que guiarán a los desarrolladores y testers.

<https://martinfowler.com/bliki/GivenWhenThen.html>  
<https://www.agilealliance.org/glossary/user-story-template/>

# Features: Behavior Driven Development



Mantener una distinción entre features e historias de usuario:

- 1.- Una Feature es una **funcionalidad del software** que se entrega a los usuarios finales o a otras partes interesadas que necesitan para alcanzar los objetivos de negocio.
- 2.- Una historia de usuario es una herramienta de **planificación** que ayuda a profundizar en los detalles de lo que se necesita para ofrecer una **característica particular**.

Es importante recordar que las historias de usuario están planeadas esencialmente como **artefactos**. Son una gran forma de **organizar el trabajo** para ofrecer una función o necesidad.

*Una vez que la Feature ha sido implementada, las historias de usuario pueden darse por terminadas.*

*Una Feature se puede construir o entregar con relativa independencia de otras Features y ser probada por los usuarios finales de forma aislada.*

*Las Features se utilizan a menudo para planificar y documentar releases.*

# Behavior Driven Development - Features

Las Features en Gherkin son **especificaciones ejecutables** y tiene dos propósitos:

1. Servir como documentación de un proyecto
2. Para pruebas automatizadas (releases).



Gherkin, es un **lenguaje** comprensible por humanos y por ordenadores, con el que vamos a describir las **funcionalidades**, definiendo el **comportamiento del software**, sin entrar en su implementación.

# Gherking

**Feature:** Indica el nombre de la funcionalidad que vamos a probar. Debe ser un título claro y explícito. Incluimos aquí una descripción en forma de historia de usuario: “**Como [rol] quiero [ característica] para que [los beneficios]**”. Sobre esta descripción empezaremos a construir nuestros escenarios de prueba.

**Scenario:** Describe cada escenario que vamos a probar.

**Given:** Provee **contexto para el escenario** en que se va a ejecutar el test, tales como puntos donde se ejecuta el test, o pre-requisitos en los datos. Incluye los pasos necesarios para poner al sistema en el estado que se desea probar.

**When:** Especifica el **conjunto de acciones que lanzan** el test. La interacción del usuario que acciona la funcionalidad que deseamos testear.

**Then:** Especifica el **resultado esperado** en el test. Observamos los cambios en el sistema y vemos si son los deseados.

```
clients.feature x
arquitecto_apis_microservicios > 01_openapi > labs > bdd > api_clients_mocks > tests > features > clients.feature
1 Feature:
2   Negocio me solicita poder consultar los clientes por su identificador principal del cliente
3
4     Scenario Outline: Obtener la información del cliente via el recurso de /clientes/versionApi
5
6       Given I set bearer token
7       When I GET 'host'/clientes/'versionApi'?idCliente=<idCliente>
8
9       Then response code should be 200
10      And response body should be valid json
11      And response body path $.id should be <id>
12      And response body path $.nombre should be <nombre>
13      And response body path $.apellidoPaterno should be <apellidoPaterno>
14      And response body path $.apellidoMaterno should be <apellidoMaterno>
15      And response body path $.edad should be <edad>
16      And response body path $.rfc should be <rfc>
17      And response body path $.email should be <email>
18      And response body path $.direccion should be <direccion>
19      And response body path $.codigoPostal should be <codigoPostal>
20      And response body path $.referencias should be <referencias>
21      And response body path $.genero should be <genero>
22 Examples:
23 |idCliente|id|nombre|apellidoPaterno|apellidoMaterno|edad|rfc|email|direccion|codigoPostal|referencias|genero|
24 |1000|1000|Yaridel|Arzate|Landa|25|CUP000825569|yaridel435@gmail.com|calle esperanza, 10 de abril, Temixco Morelos|62588|Frente a la base de camiones LOS PEPES|F|
25 |5000|5000|Jovani|Arzate|Cabrera|28|456787656453454sds|jovaniac@gmail.com|bugambilias 2, colonia 10 de abril, Temixco Morelos|0634560|enfrente de la tortilleria ALFONSOS|M|
```

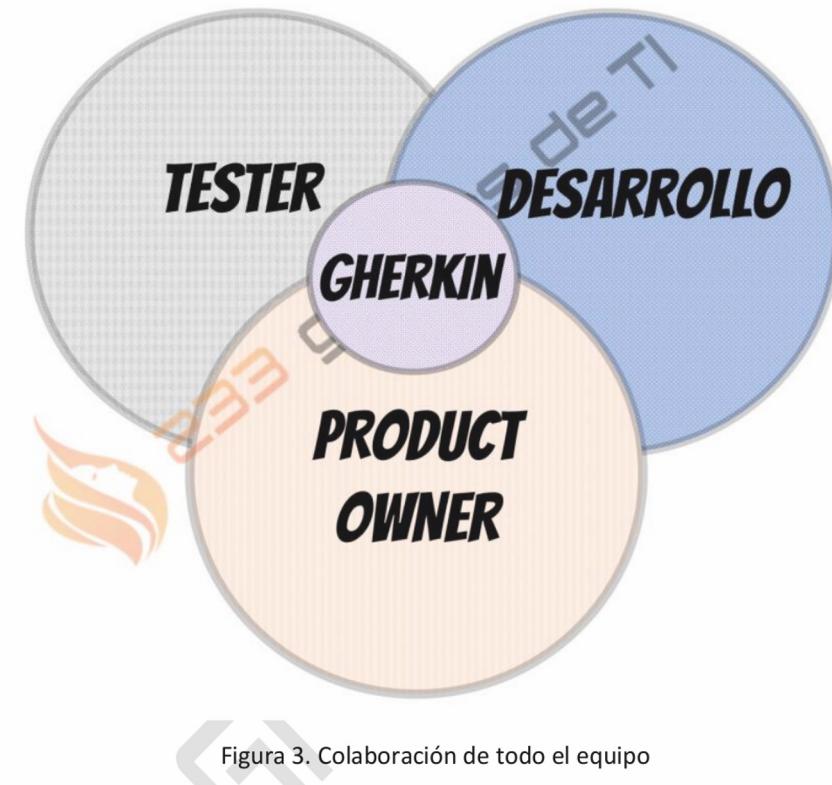
Las Features en Gherkin se escriben en ficheros de texto plano con la extensión “.feature”

Lo normal es probar **distintos escenarios para comprobar una determinada funcionalidad**, de esta forma vamos a pasar de nuestras historias de usuario a pruebas de comportamiento automatizables.

Para automatizar estas pruebas necesitamos apoyarnos en herramientas, como **Cucumber, que entiende perfectamente el lenguaje Gherkin**.

# Historias de usuario

Colaboración: 3 AMIGOS



# Historias de usuario

Colaboración: 3 AMIGOS

Desarrollo tradicional



Figura 4. Comunicación en un desarrollo en cascada

# Historias de usuario

Colaboración: 3 AMIGOS

Desarrollo ágil



Figura 5. Comunicación en un desarrollo ágil con BDD

# BDD – Features syntaxis

'Given-When-Then' como lenguaje común con BDD

Para definir los casos BDD para una historia de usuario se deben definir bajo el patrón '**Given-When-Then**', que se define como:

Given 'dado': Se especifica el escenario, las **precondiciones**.

When 'cuando': Las **condiciones de las acciones** que se van a ejecutar.

Then 'entonces': **El resultado esperado**, las validaciones a realizar.

Un ejemplo práctico sería:

Given: Dado que el usuario no ha introducido ningún dato en el formulario.

When: Cuando hace clic en el botón Enviar.

Then: Se deben mostrar los mensajes de validación apropiados.

'Role-Feature-Reason' como lenguaje común con BDD

Este patrón también se utiliza en BDD para ayudar a la creación de historias de usuarios. Esta se define como:

As a 'Como': Se especifica el tipo de usuario.

I want 'deseo': Las necesidades que tiene.

So that 'para que': Las características para cumplir el objetivo.

Un ejemplo práctico de historia de usuario sería: - Como cliente interesado, deseo ponerme en contacto mediante el formulario, para que atiendan mis necesidades.

# BDD - Cucumber

**Cucumber** es una más de las herramientas software (escrita en Ruby) que se utilizan en BDD para **convertir los Scenarios escritos en Gherkin** en el esqueleto de una prueba o en la definición de un paso (**StepDefinition**).

Cuando Cucumber ejecuta un **Scenario**, si el sistema se **comporta como se ha descrito en él**, se dice que el Scenario se ha pasado, es decir, el Scenario ha tenido éxito (**succeed**) , **si no ocurre, entonces el Scenario ha fallado (failed)**.

*Cada vez que se añada un nuevo Scenario y al ejecutar Cucumber, lo marque como pasado, se habrá añadido una nueva funcionalidad al sistema.*

Los **Steps Definitions** (o definición de los pasos o esqueleto del caso de prueba) son la transformación automática por parte de Cucumber de los **Scenarios de Gherkin**.



```
15 scenarios (15 passed)
182 steps (182 passed)
0m00.390s
[02:35:22] Finished 'cucumber-test' after 733 ms
[02:35:22] Starting 'plantilla-html'...
Cucumber HTML report features/resultados/cucumber_report.html generated successfully.

OK: 0 assertions (39ms)
[02:35:22] Finished 'plantilla-html' after 50 ms
```

# apickli-gherkin.js

**Apickli** es un framework de **pruebas de integración** de APIs REST,  
basado en **cucumber.js**.

Proporciona un framework de cucumber y una colección de  
funciones de utilidad para que las pruebas de las APIs sean fáciles y  
requieren menos tiempo.



# Hands-ON API de Clientes Features

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training\\_apis\\_apigee](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training_apis_apigee)

[enero 2024/01 openapi/lab4 ejercicio/01 openapi/lab4/bdd/api clientes mocks](#)

```
145      </a>
146    </li>
147  </ul>
148 </div>
149  );
150}
151
152  renderWhatsNewLinks() {
153    return (
154      <div className={styles.whatsNew}>
155        <h4 className={styles.whatsNewTitle}>
156          <ul className={styles.whatsNewList}>
157            {this.renderWhatIsNewItem(
158              {this.renderWhatIsNewItem(
159                {this.renderWhatIsNewItem(
160                  {this.renderWhatIsNewItem(
161                    {this.renderWhatIsNewItem(
162                      {this.renderWhatIsNewItem(
163                        {this.renderWhatIsNewItem(
164                          {this.renderWhatIsNewItem(
165                            <li>
166                              <div>
167                                <ul>
168                                  <li>
169                                    renderWhatsNewItem(title, url)
170                                  return (
171                                    <li className={styles.footerList}>
172                                      <a href={url} onClick={trackUrl(url)}>
173                                        <div>
174                                          <img alt="Unsplash logo" />
175                                          <span>Open in new window</span>
176                                        </div>
177                                      </a>
178                                    </li>
179                                );
180                            </ul>
181                          );
182                        );
183                      );
184                    );
185                  );
186                );
187              );
188            );
189          );
190        );
191      );
192    );
193  }
194
195  renderFooterSub() {
196    return (
197      <div className={styles.footerSub}>
198        <a href="/" title="Home - Unsplash">
199          <Icon type="logo" className={styles.footerSubLogo}>
200            </Icon>
201          </a>
202        </div>
203        <span className={styles.footerSlogan}>
204          <img alt="Unsplash logo" />
205          <span>A free community of creative people</span>
206        </span>
207      );
208    );
209  }
210
211  render() {
212    return (
213      <footer className={styles.footerGlobal}>
214        <div className="container">
215          {this.renderFooterMain()}
216          {this.renderFooterSub()}
217        </div>
218      </footer>
219    );
220  }
221}
```

# Hands-ON API de Créditos Features

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training\\_apis\\_apigee\\_enero\\_202401\\_openapi/lab4\\_ejercicios/01\\_openapi/lab6/bdd/api\\_creditosMocks](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training_apis_apigee_enero_202401_openapi/lab4_ejercicios/01_openapi/lab6/bdd/api_creditosMocks)

```
145      </a>
146    </li>
147  </ul>
148 </div>
149  );
150}
151
152  renderWhatsNewLinks() {
153    return (
154      <div className={styles.whatsNew}>
155        <h4 className={styles.whatsNewTitle}>
156          <ul className={styles.whatsNewList}>
157            {this.renderWhatIsNewItem(
158              {this.renderWhatIsNewItem(
159                {this.renderWhatIsNewItem(
160                  {this.renderWhatIsNewItem(
161                    {this.renderWhatIsNewItem(
162                      {this.renderWhatIsNewItem(
163                        {this.renderWhatIsNewItem(
164                          {this.renderWhatIsNewItem(
165                            <li>
166                              <a href="#">
167                                {title}
168                              </a>
169                            </li>
170                          );
171                        );
172                      );
173                    );
174                  );
175                );
176              );
177            );
178          );
179        );
180      </ul>
181    );
182  }
183
184  renderFooterSub() {
185    return (
186      <div className={styles.footerSub}>
187        <Link to="/" title="Home - Unsplash">
188          <Icon type="logo" className={styles.footerSubLogo}>
189            />
190          </Link>
191        <span className={styles.footerSlogan}>
192          <img alt="Unsplash logo" />
193        </span>
194      </div>
195    );
196  }
197
198  render() {
199    return (
200      <footer className={styles.footerGlobal}>
201        <div className="container">
202          {this.renderFooterMain()}
203          {this.renderFooterSub()}
204        </div>
205      </footer>
206    );
207  }
208}
```

# Hands-ON API de Empleados Features Homework

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training\\_apis\\_apigee](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training_apis_apigee)

enero 2024/lab4 homework/bdd/api empleados mock

```
userDetailsCardOnHover = showOnHover(UserDetailsCard);

UserLink = {
  ...
  secondaryLink,
  children,
  userAvatar,
  name,
  ...
  render() {
    return (
      <div className={styles.container}>
        {includeAvatar && (
          <UserDetailsCardOnHover
            user={user}
            delay={CARD_HOVER_DELAY}
            wrapperClassName={styles.avatarContainer}>
            <Avatar ... />
          </UserDetailsCardOnHover>
        )}
        <div
          className={classNames(styles.list, styles.inline, styles.listContainer)}
        >
          <UserDetailsCardOnHover user={user} delay={CARD_HOVER_DELAY}>
            <Link
              to={({ primary, secondary }) => {
                const type = primary ? 'primary' : 'secondary';
                const alt = primary ? 'secondaryLink' : 'inlineLink';
                const activeClassName = primary ? 'active' : '';
                return (
                  <a href={primary} alt={alt} type={type}>
                    {children || user.name}
                  </a>
                );
              }}
            >
              {!secondaryLink ? null : <a href={secondary}>
                <span className={classNames(styles.listItem, activeClassName)}>
                  {secondaryLink.label}
                </span>
              </a>
            >}
          </UserDetailsCardOnHover>
        </div>
        <span>
          ...
        </span>
      </div>
    );
  }
};

renderWhatsNewLinks() {
  return (
    <div className={styles.whatsNew}>
      <h4>What's New</h4>
      <ul className={styles.whatsNewList}>
        {this.renderWhatIsNewItem('New Item 1')}
        {this.renderWhatIsNewItem('New Item 2')}
        {this.renderWhatIsNewItem('New Item 3')}
        {this.renderWhatIsNewItem('New Item 4')}
        {this.renderWhatIsNewItem('New Item 5')}
      </ul>
    </div>
  );
}

renderWhatIsNewItem(title, url) {
  return (
    <li className={styles.whatsNewItem}>
      <a href={url} target={trackUrl ? '_self' : '_blank'} rel="noopener noreferrer">
        <span>{title}</span>
      </a>
    </li>
  );
}

renderFooterSub() {
  return (
    <div className={styles.footerSub}>
      <Link to="/" title="Home - Unsplash">
        <Icon type="logo" className={styles.footerSubLogo}>
        </Icon>
      </Link>
      <span className={styles.footerSlogan}>...
      </span>
    </div>
  );
}

render() {
  return (
    <footer className={styles.footerGlobal}>
      <div className="container">
        {this.renderFooterMain()}
        {this.renderFooterSub()}
      </div>
    </footer>
  );
}
```

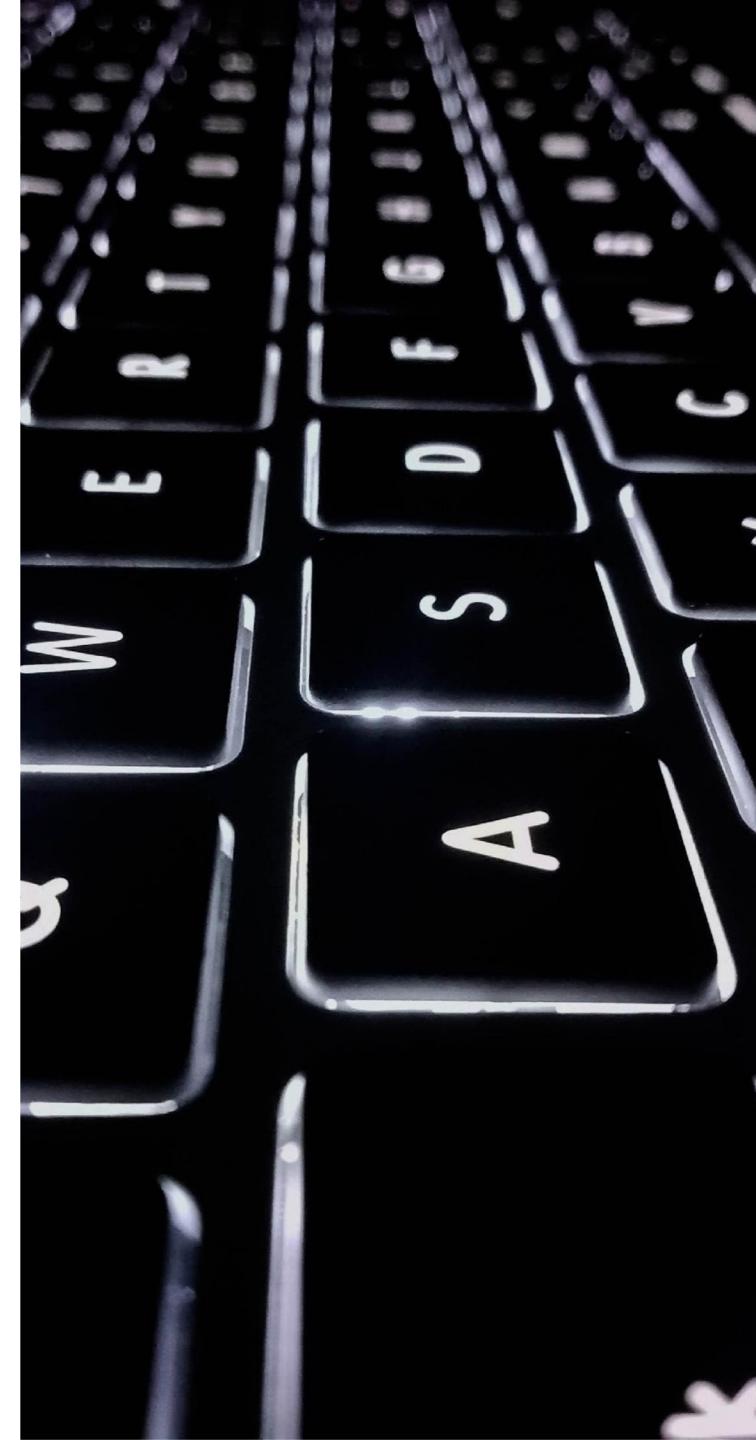
# Hands-ON API de Seguros Crédito Homework

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training\\_apis\\_apigee\\_enero\\_2024/01/openapi/lab4/ejercicio/01/openapi/lab6/bdd/api\\_segurosMocks](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training_apis_apigee_enero_2024/01/openapi/lab4/ejercicio/01/openapi/lab6/bdd/api_segurosMocks)

```
145      </a>
146    </li>
147  </ul>
148 </div>
149  );
150}
151
152  renderWhatsNewLinks() {
153    return (
154      <div className={styles.whatsNew}>
155        <h4 className={styles.whatsNewTitle}>
156          <ul className={styles.whatsNewList}>
157            {this.renderWhat(<li>)}
158            {this.renderWhat(<li>)}
159            {this.renderWhat(<li>)}
160            {this.renderWhat(<li>)}
161            {this.renderWhat(<li>)}
162            {this.renderWhat(<li>)}
163            {this.renderWhat(<li>)}
164          </ul>
165        </div>
166      );
167    }
168
169  renderFooterMain() {
170    return (
171      <div className={styles.footer}>
172        <ul className={styles.footerList}>
173          <li className={styles.footerItem}>
174            <a href={trackUrl(url)} target="_blank" rel="noopener noreferrer">
175              {title}
176            </a>
177          </li>
178        </ul>
179      </div>
180    );
181  }
182
183  renderFooterSub() {
184    return (
185      <div className={styles.footerSub}>
186        <Link to="/" title="Home - Unsplash" type="logo" className={styles.footerSubLogo}>
187          <Icon />
188        </Link>
189      </div>
190    );
191  }
192
193  render() {
194    return (
195      <footer className={styles.footerGlobal}>
196        <div className="container">
197          {this.renderFooterMain()}
198          {this.renderFooterSub()}
199        </div>
200      </footer>
201    );
202  }
203
204  renderFooterMain() {
205    return (
206      <div>
```

# Módulo I

## Deploy de APIs primeros pasos con Docker



# Desplegar API con Docker

Tener un ambiente local con nuestras APIs expuestas vía MOCKs esta bien, pero cuando nuestro desarrollo solo es de nosotros, normalmente al crear nuestras APIs y realizar una integración queremos exponerlas a terceros o diferentes equipos de desarrollo, por lo cual es necesario que entreguemos nuestro ambiente de manera sencilla, homogénea y ephemera para poder desplegar n veces tantos cambios sufra nuestra API mientras el backend es construido a la vez.

Cuenta en dockerHub

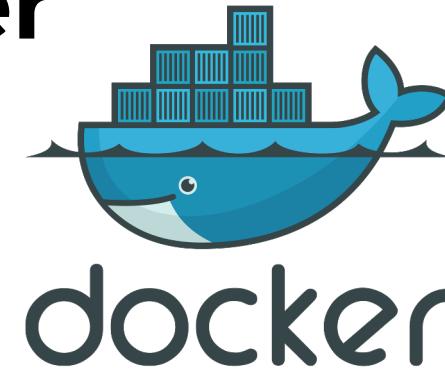
Tener instalado el engine de docker

Hacer login a dockerhub

1.- docker login

2.- docker build -t jovaniac/api-clientes-certificatic:0.1

3.- docker run -i -t -p 8000:8000 jovaniac/api-clientes-certificatic:0.1



```
openapi/lab8/api_clientes_deploy$ docker build -t jovaniac/api-clientes-certificatic:0.1 .
Sending build context to Docker daemon 136.7kB
Step 1/9 : FROM node:8
8: Pulling from library/node
146bd6a88618: Downloading [=====] 41.83MB/45.38MB
9935d0c62ace: Download complete
db0efb86e806: Download complete
e705a4c4fd31: Downloading [=====] 43.14MB/50.07MB
c877b722db6f: Downloading [=====] 37.13MB/214.8MB
645c20ec8214: Pulling fs layer
db8fdbd9db2fe: Waiting
1c151cd1b3ea: Pulling fs layer
fb993995f40: Waiting
```

```
Status: Downloaded newer image for node:8
--> 8eedad13757f4
Step 2/9 : WORKDIR /api
--> Running in ldc0f1bb6083
Removing intermediate container ldc0f1bb6083
--> c6c18f6d9752
Step 3/9 : RUN mkdir -p /api/node /api/node/mock/clientes
--> Running in f918ba950d7b
Removing intermediate container f918ba950d7b
--> e4fb5b93f9af
Step 4/9 : RUN chmod 777 -R /api
--> Running in 954ceal16c888
Removing intermediate container 954ceal16c888
--> de18e46a8339
Step 5/9 : COPY ./node/config-generated.json /api/node
--> 066715961062
Step 6/9 : COPY ./node/mock/clientes/*.json /api/node/mock/clientes/
--> 53c2541562cf
Step 7/9 : RUN npm install -g apimocker --unsafe-perm=true --allow-root
--> Running in 3ec57ff6aaaf3
/usr/local/bin/apimocker -> /usr/local/lib/node_modules/apimocker/bin/apimocker.js
> jsonpath@1.0.2 postinstall /usr/local/lib/node_modules/apimocker/node_modules/jsonpath
> node lib/aesprim.js > generated/aesprim-browser.js

+ apimocker@0.1.4
added 109 packages from 172 contributors in 7.93s
Removing intermediate container 3ec57ff6aaaf3
--> dc907f3406a6
Step 8/9 : EXPOSE 8000
--> Running in baa0f3640efd
Removing intermediate container baa0f3640efd
--> 2d83ddce0403
Step 9/9 : CMD ["apimocker", "-c", "/api/node/config-generated.json", "-p", "8000"]
--> Running in 61522ed92876
Removing intermediate container 61522ed92876
--> 3c20376eae84
Successfully built 3c20376eae84
Successfully tagged jovaniac/api-clientes-certificatic:0.1
```

A screenshot of the Postman application interface. At the top, there's a navigation bar with tabs for 'Filter', 'History', 'Collections' (which is selected), 'APIs BETA', and 'Trash'. Below the navigation, there's a search bar and a 'New Collection' button. The main area shows a collection named 'api clientes\_mock' containing 13 requests. One request is highlighted: 'POST API Clientes post 201'. The 'Body' tab is selected, showing a JSON payload with fields like 'nombre', 'Varidel', 'apellidoPaterno', 'Landa', 'rfc', 'correoElectronico', 'direccion', 'telefonoPostal', 'referencias', and 'genero'. The 'Headers' tab shows 'Content-Type: application/json'. Other tabs include 'Params', 'Authorization', and 'Tests'. To the right, there are sections for 'Launchpad', 'POST API Clientes post 201' (with preview and details), and 'Test Results'.

# Desplegar API en Kubernetes

Tán tán tán tán...

Cuenta en dockerHub

Tener instalado el engine de docker

Hacer login a dockerhub

1.- docker login

2.- docker build -t jovaniac/api-clientes-certificatic:0.1 .

3.- docker run -i -t -p 8000:8000 jovaniac/api-clientes-certificatic:0.1

4.- docker push jovaniac/api-clientes-certificatic:0.1

## Cuenta de gmail

5.- Activar el modo gratuito de 300USD para el uso de GCP

6.- Crear un cluster de Kubernetes, el más pequeño

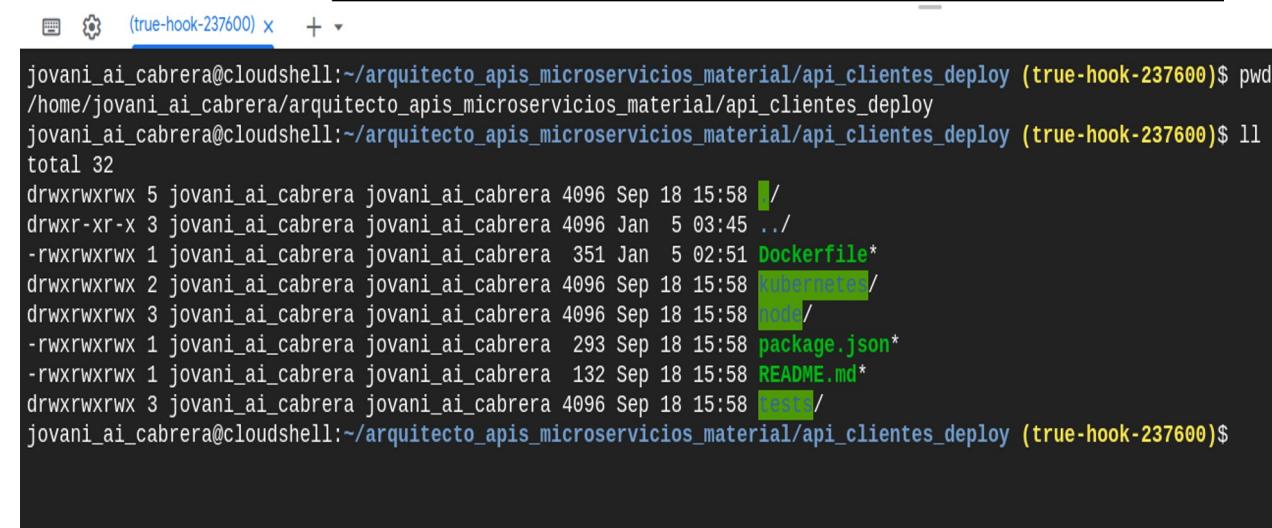
7.- Configurar kubectl

8.- Subir archivos manifiestos del deployment y service de kubernetes en la consola de GCP

9.- zip api\_clientes\_deploy.zip api\_clientes\_deploy/

kubectl apply -f /kubernetes

```
openapi/lab8/api_clientes_deploy$ docker push jovaniac/api-clientes-certificatic:0.1
The push refers to repository [docker.io/jovaniac/api-clientes-certificatic]
ffa32107163d: Pushing [=====] 12.44MB
9d0c1aa1b39a: Pushed
12fbe27e951: Pushed
bda06ec3e56d: Pushed
03949bc08d03: Pushed
6b70a79cedc2: Pushed
423451ed44f2: Mounted from library/node
b2aaef85d6633: Mounted from library/node
88601a85c11: Mounted from library/node
42f9c2f9c08e: Mounted from library/node
99e8bd3efaa: Mounted from library/node
beee1e39d7c3a: Preparing
1f59a4b2e206: Waiting
0ca7f54856c0: Waiting
ebb9ae013834: Waiting
```



A screenshot of a terminal window titled '(true-hook-237600) x'. The window shows two commands being run. The first command is 'pwd' which outputs the current working directory as '/home/jovani\_ai\_cabrera/arquitecto\_apis\_microservicios\_material/api\_clientes\_deploy'. The second command is 'll' which lists files in the directory. The output shows several files and directories including 'Dockerfile\*', 'kubernetes/\*', 'node/\*', 'package.json\*', 'README.md\*', and 'tests/\*'. The file 'Dockerfile\*' is highlighted in green.

```
jovani_ai_cabrera@cloudshell:~/arquitecto_apis_microservicios_material/api_clientes_deploy (true-hook-237600)$ pwd
/home/jovani_ai_cabrera/arquitecto_apis_microservicios_material/api_clientes_deploy
jovani_ai_cabrera@cloudshell:~/arquitecto_apis_microservicios_material/api_clientes_deploy (true-hook-237600)$ ll
total 32
drwxrwxrwx 5 jovani_ai_cabrera jovani_ai_cabrera 4096 Sep 18 15:58 /
drwxr-xr-x 3 jovani_ai_cabrera jovani_ai_cabrera 4096 Jan 5 03:45 ../
-rwxrwxrwx 1 jovani_ai_cabrera jovani_ai_cabrera 351 Jan 5 02:51 Dockerfile*
drwxrwxrwx 2 jovani_ai_cabrera jovani_ai_cabrera 4096 Sep 18 15:58 kubernetes/
drwxrwxrwx 3 jovani_ai_cabrera jovani_ai_cabrera 4096 Sep 18 15:58 node/
-rwxrwxrwx 1 jovani_ai_cabrera jovani_ai_cabrera 293 Sep 18 15:58 package.json*
-rwxrwxrwx 1 jovani_ai_cabrera jovani_ai_cabrera 132 Sep 18 15:58 README.md*
drwxrwxrwx 3 jovani_ai_cabrera jovani_ai_cabrera 4096 Sep 18 15:58 tests/
jovani_ai_cabrera@cloudshell:~/arquitecto_apis_microservicios_material/api_clientes_deploy (true-hook-237600)$
```

# Desplegar API en Kubernetes

The screenshot shows the Google Cloud Platform interface for creating a Kubernetes cluster. The cluster is named 'your-first-cluster-1'. It is set to use a 'Zona' (Region) and has a 'us-central1-a' zone selected. The master version is '1.15.4-gke.22'. The configuration includes a 'Clúster estándar' (Standard Cluster) with 'Autoscalado' (Autoscaling) and 'Stackdriver Logging y Monitoring' (Stackdriver Logging and Monitoring) enabled. The disk size is 30GB. A note indicates that Compute Engine pricing will apply to the single node.

```
gcloud container clusters get-credentials your-first-cluster-1 --zone us-central1-a --  
project true-hook-237600
```

```
jovani_ai_cabrera@cloudshell:~/arquitecto_apis_microservicios_material/api_clientes_deploy (true-hook-237600)$  
jovani_ai_cabrera@cloudshell:~/arquitecto_apis_microservicios_material/api_clientes_deploy (true-hook-237600)$ gcloud container clusters get-credentials your-first-cluster-1 --zone us-central1-a --project true-hook-237600  
Fetching cluster endpoint and auth data.  
kubeconfig entry generated for your-first-cluster-1.  
jovani_ai_cabrera@cloudshell:~/arquitecto_apis_microservicios_material/api_clientes_deploy (true-hook-237600)$
```

```
kubectl apply -f kubernetes/
```

```
jovani_ai_cabrera@cloudshell:~/arquitecto_apis_microservicios_material/api_clientes_deploy (true-hook-237600)$ kubectl apply -f kubernetes/  
deployment.extensions/api-clientes created  
service/api-clientes created  
jovani_ai_cabrera@cloudshell:~/arquitecto_apis_microservicios_material/api_clientes_deploy (true-hook-237600)$
```

```
kubectl get pods,svc --all-namespaces
```

```
jovani_ai_cabrera@cloudshell:~/arquitecto_apis_microservicios_material/api_clientes_deploy (true-hook-237600)$ kubectl get pods,svc --all-namespaces  
NAMESPACE     NAME                               READY   STATUS    RESTARTS   AGE  
default       pod/api-clientes-6cf77f49db-zt4ms   1/1    Running   0          100s  
kube-system   pod/kube-dns-65678995b-198mt      4/4    Running   0          4m56s  
kube-system   pod/kube-dns-autoscaler-6d7c4b8447-nxf17 1/1    Running   0          4m44s  
kube-system   pod/kube-proxy-gke-your-first-cluster-1-pool-1-2b760ed3-p271 1/1    Running   0          4m22s  
kube-system   pod/l7-default-backend-84c9fcbb-7g85m   1/1    Running   0          4m55s  
kube-system   pod/metrics-server-v0.3.3-85dfcbb78-cnbfr 2/2    Running   0          4m15s  
  
NAMESPACE     NAME           TYPE        CLUSTER-IP   EXTERNAL-IP  PORT(S)   AGE  
default       service/api-clientes   LoadBalancer  10.0.9.89   35.184.96.225  8000:32507/TCP  100s  
default       service/kubernetes  ClusterIP    10.0.0.1    <none>     443/TCP   5m21s  
kube-system   service/default-http-backend  NodePort    10.0.2.48   <none>     80:32604/TCP  4m55s  
kube-system   service/kube-dns   ClusterIP    10.0.0.10   <none>     53/UDP,53/TCP  4m56s  
kube-system   service/metrics-server  ClusterIP    10.0.9.60   <none>     443/TCP   4m52s  
jovani_ai_cabrera@cloudshell:~/arquitecto_apis_microservicios_material/api_clientes_deploy (true-hook-237600)$
```

# Desplegar API en Kubernetes

The screenshot shows a Postman interface with the following details:

- Header:** POST API Clientes post 201 2
- Method:** POST
- URL:** http://35.184.96.225:8000/clientes/v1
- Params:** Authorization, Headers (9), Body (selected), Pre-request Script, Tests, Settings
- Query Params:** Key, Value
- Body:** Body (selected), Cookies, Headers (13), Test Results
- JSON View:** Pretty, Raw, Preview, Visualize BETA, JSON (selected)

```
1 {
2   "mensaje": "El Cliente se ha creado exitosamente",
3   "codigo": "201",
4   "detalles": {
5     "id": "2000",
6     "nombre": "Jovani",
7     "apellidoPaterno": "Arzate",
8     "apellidoMaterno": "Landa",
9     "edad": "26",
10    "rfc": "456787656453454sds",
11    "email": "jovaniac@gmail.com",
12    "direccion": "bugambilias 2, colonia 10 de abril, Temixco Morelos",
13    "codigoPostal": "0634560",
14    "referencias": "enfrente de la tortilleria ALFONSOS",
15    "genero": "M"
16  }
17 }
```

NAMESPACE	NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
default	service/api-clientes	LoadBalancer	10.0.9.89	35.184.96.225	8000:32507/TCP	100s
default	service/kubernetes	ClusterIP	10.0.0.1	<none>	443/TCP	5m21s
kube-system	service/default-http-backend	NodePort	10.0.2.48	<none>	80:32604/TCP	4m55s
kube-system	service/kube-dns	ClusterIP	10.0.0.10	<none>	53/UDP, 53/TCP	4m56s
kube-system	service/metrics-server	ClusterIP	10.0.9.60	<none>	443/TCP	4m52s

jovani\_ai\_cabrera@cloudshell:~/arquitecto\_apis\_microservicios\_material/api\_clientes\_deploy (true-hook-237600)\$

# Hands-ON API Mocker + Docker + Deploy Kubernetes

```
userDetailsCardOnHover = showOnHover(UserDetailsCard);

UserLink = {
  ...
  secondaryLink,
  ...
  dren,
  ...
  udeAvatar,
  ...
  ne,
  ...
  in className={styles.container}>
    ...
    includeAvatar && (
      ...
      <UserDetailsCardOnHover
        user={user}
        delay={CARD_HOVER_DELAY}
        wrapperClassName={styles.avatarContainer}>
        ...
        <Avatar
          ...
        />
      </UserDetailsCardOnHover>
    )
  ...
  <div
    ...
    className={classNames(
      styles.lineContainer,
      inline & styles.inlineContainer
    )}
  >
    ...
    <UserDetailsCardOnHover user={user} delay={CARD_HOVER_DELAY}>
      ...
      <Link
        to={({ path }) => buildUserUrl(path)}
        ...
        className={classNames(
          styles.inlineLink,
          [styles.centerXname] ? styles.centerXlink,
          [styles.inlineLink] ? inline,
        )}
      >
        ...
        {children || user.name}
      </Link>
    ...
    & !secondaryLink
    ? null
    : <a
        href={secondaryLink.href}
        ...
        className={classNames(styles.name, [
          ...
        ])}
      >
        ...
        {secondaryLink.label}
      </a>
    ...
  </UserDetailsCardOnHover>
</div>

145   </a>
146   </li>
147   </ul>
148   </div>
149   );
150 }
151
152 renderWhatsNewLinks() {
153   return (
154     <div className={styles.whatsNew}>
155       <h4 className={style}>
156         <ul className={classNames(
157           ...
158           {this.renderWhat
159           ...
160           {this.renderWhat
161           ...
162           {this.renderWhat
163           ...
164           {this.renderWhat
165           ...
166           {this.renderWhat
167           ...
168           {this.renderWhat
169         </ul>
170       </div>
171     );
172   );
173   ...
174   ...
175   ...
176   ...
177   ...
178   ...
179   ...
180   ...
181   ...
182   ...
183   ...
184   ...
185   ...
186   ...
187   ...
188   ...
189   ...
190   ...
191   ...
192   ...
193   ...
194   ...
195   ...
196   ...
197   ...
198   ...
199   ...
200   ...
201   ...
202   ...
203   ...
204   ...
205   ...
206   ...
);
return (
  <div className={styles.footerGlobal}>
    <div className="container">
      {this.renderFooterMain()}
      {this.renderFooterSub()}
      ...
    </div>
  </div>
);
}

```

<https://git-codecommit.us-east-1.amazonaws.com/v1/repos/training/apis/apigee/enero/2024/lab8/>



## Q&A

# Módulo II API Gateway **apigee**