Aplicaciones Distribuidas en Internet

Tema 2:

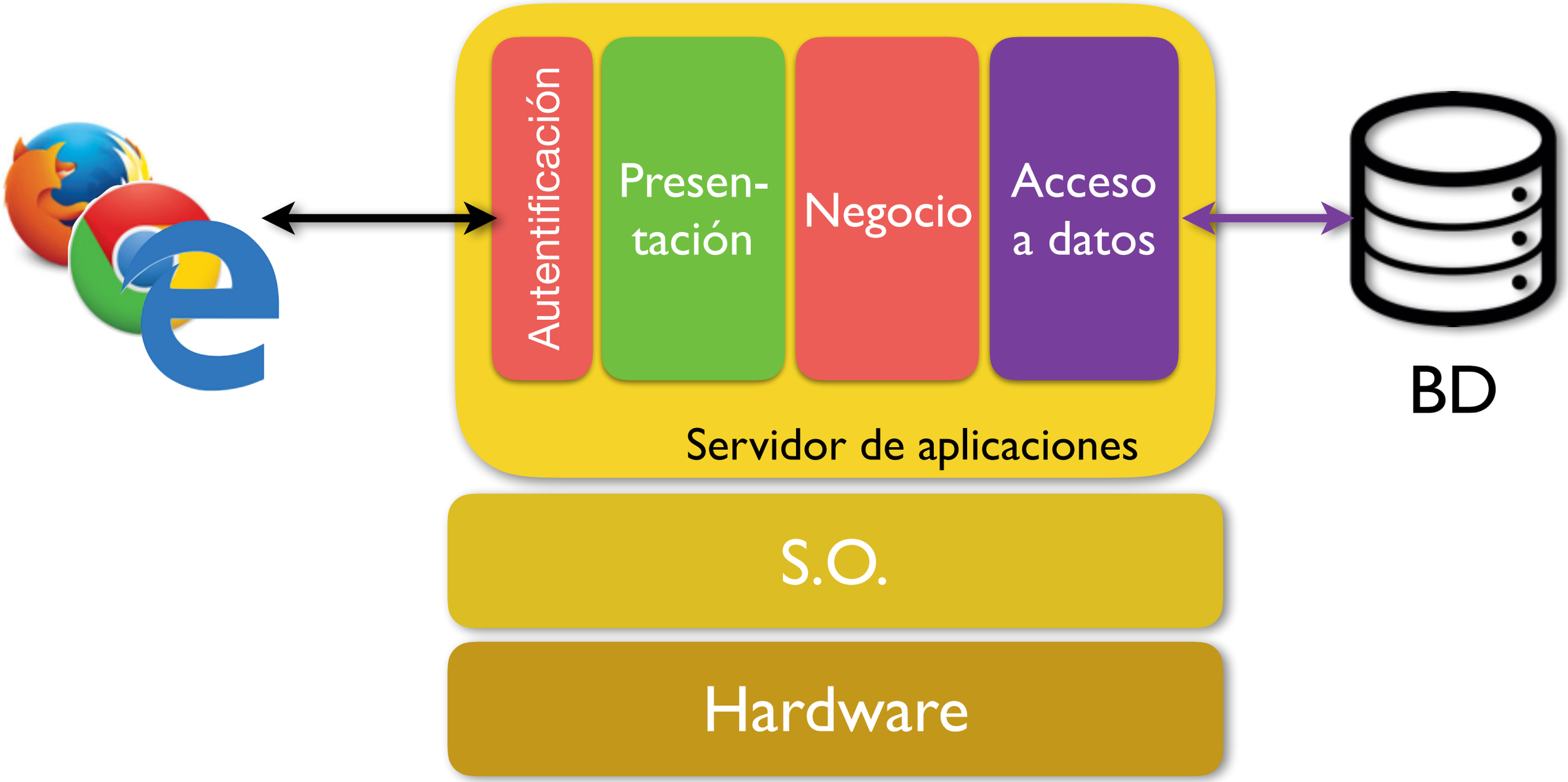
Aplicaciones web “en la nube”

Tema 2: Aplicaciones web en “la nube”

2.1 Introducción: IaaS vs PaaS vs BaaS vs FaaS vs ...

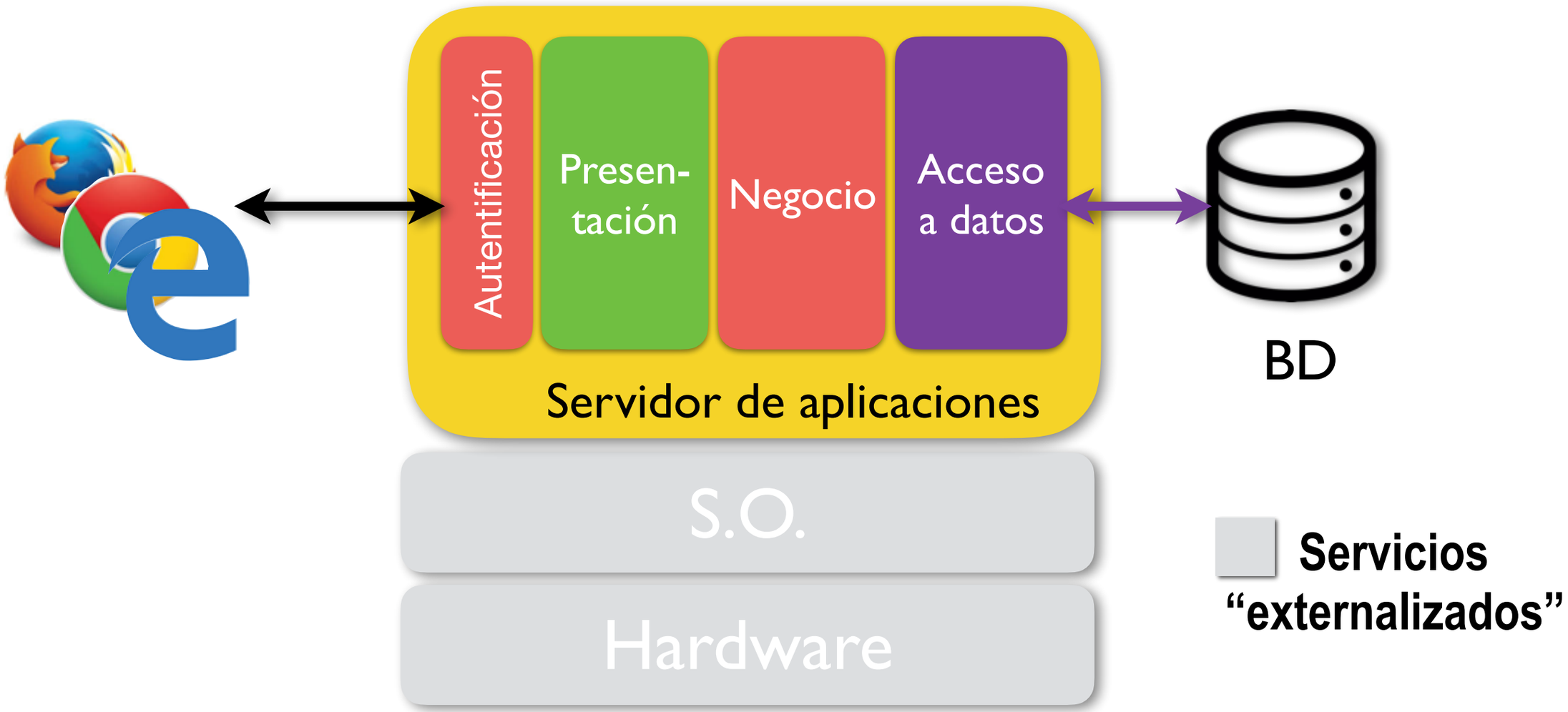
Aplicación clásica sin "nube"

Es nuestra responsabilidad no solo la **aplicación** sino también la **infraestructura**



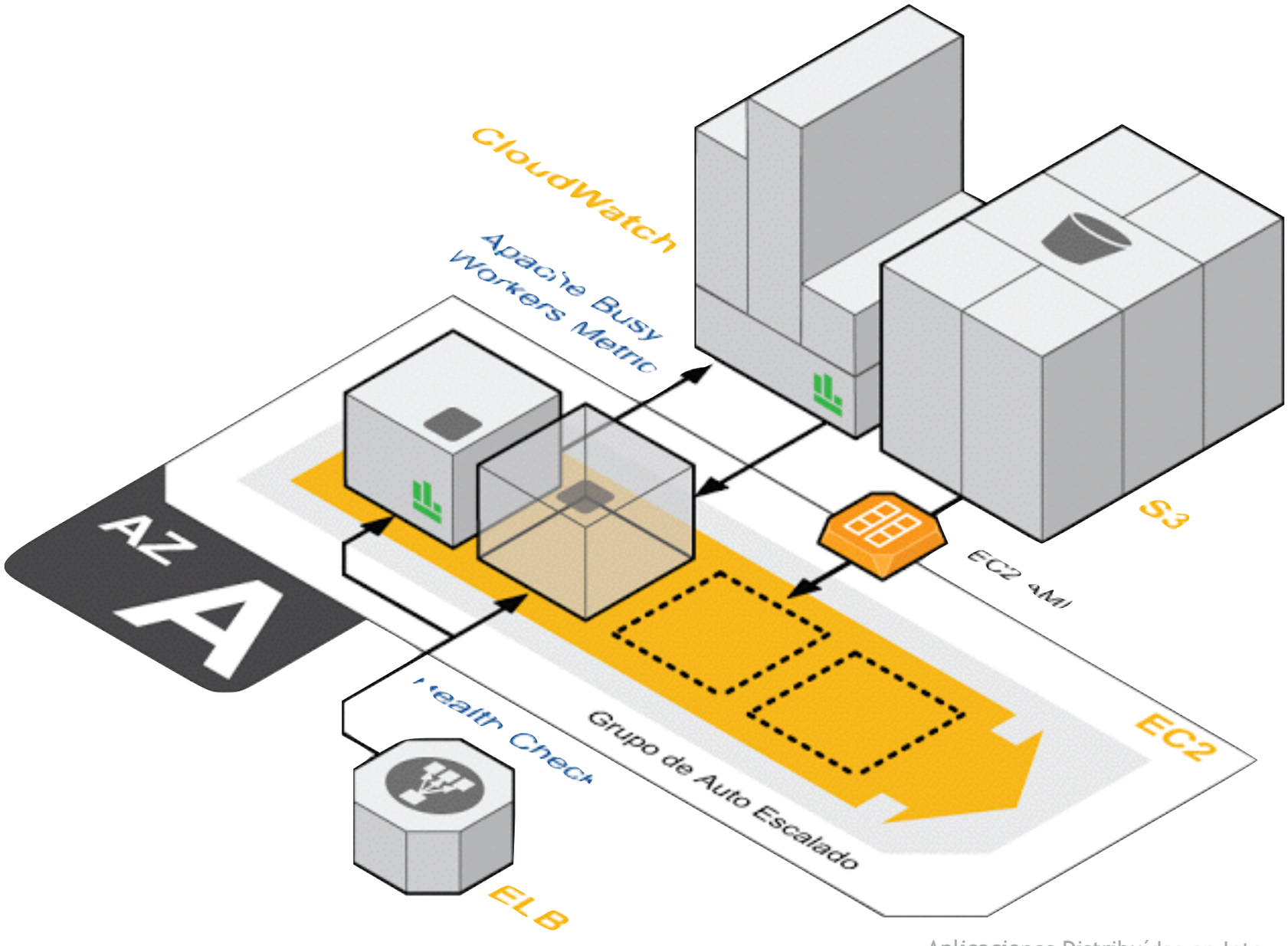
IaaS (Infrastructure as a Service)

- Se externaliza la **Infraestructura básica (Hardware + S.O.)**, aunque normalmente se usan máquinas virtuales. Seguimos teniendo que instalar/gestionar el software necesario (servidor y BD)
- Ejemplos: Amazon EC2, Azure, Rackspace,…



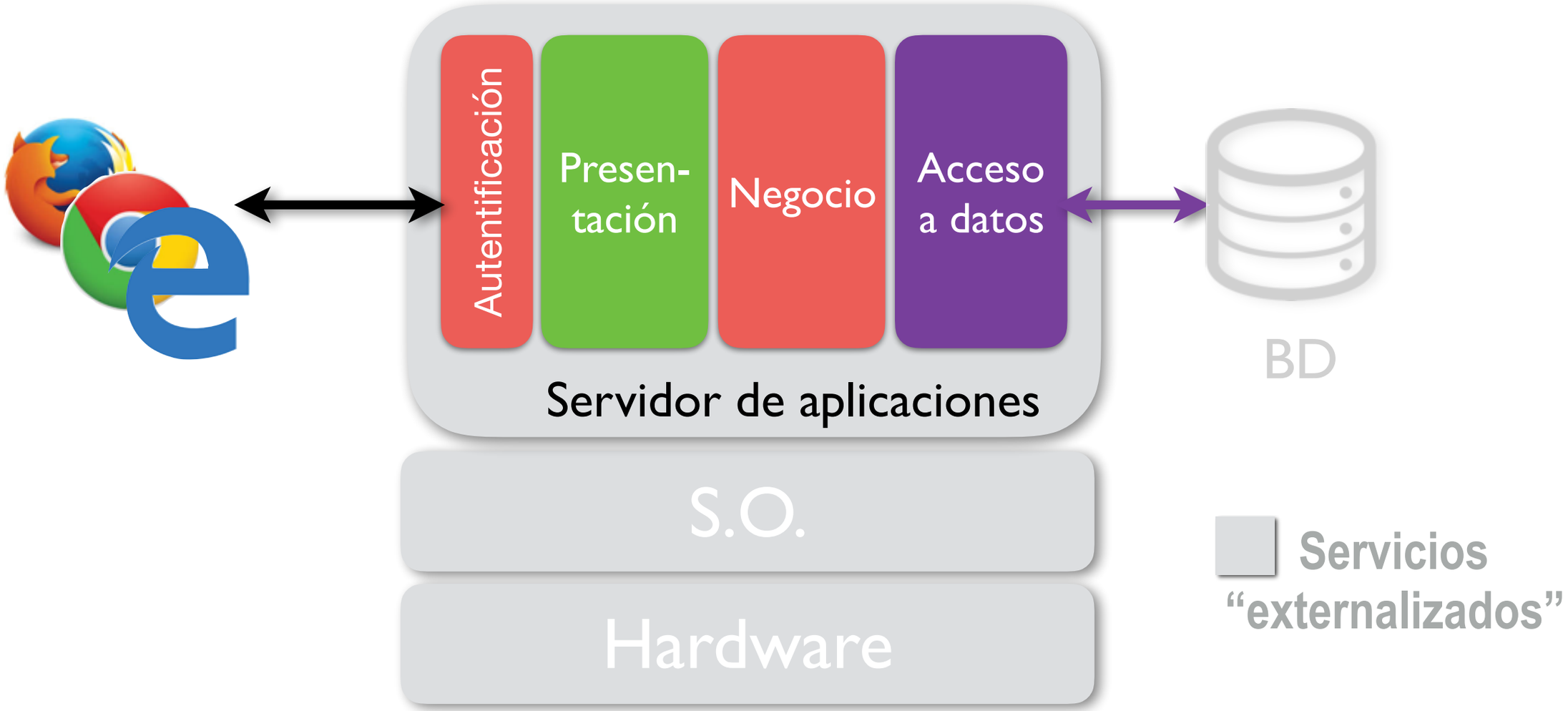
Servicios adicionales de IaaS

- Balanceadores de carga
- Autoescalado
- …



PaaS (Platform as a Service)

- IaaS y además **soporte para ejecutar aplicaciones web** en distintos lenguajes: servidores web, bases de datos, …
- Ejemplos: AWS Elastic Beanstalk, Azure, Heroku, Google App Engine, …



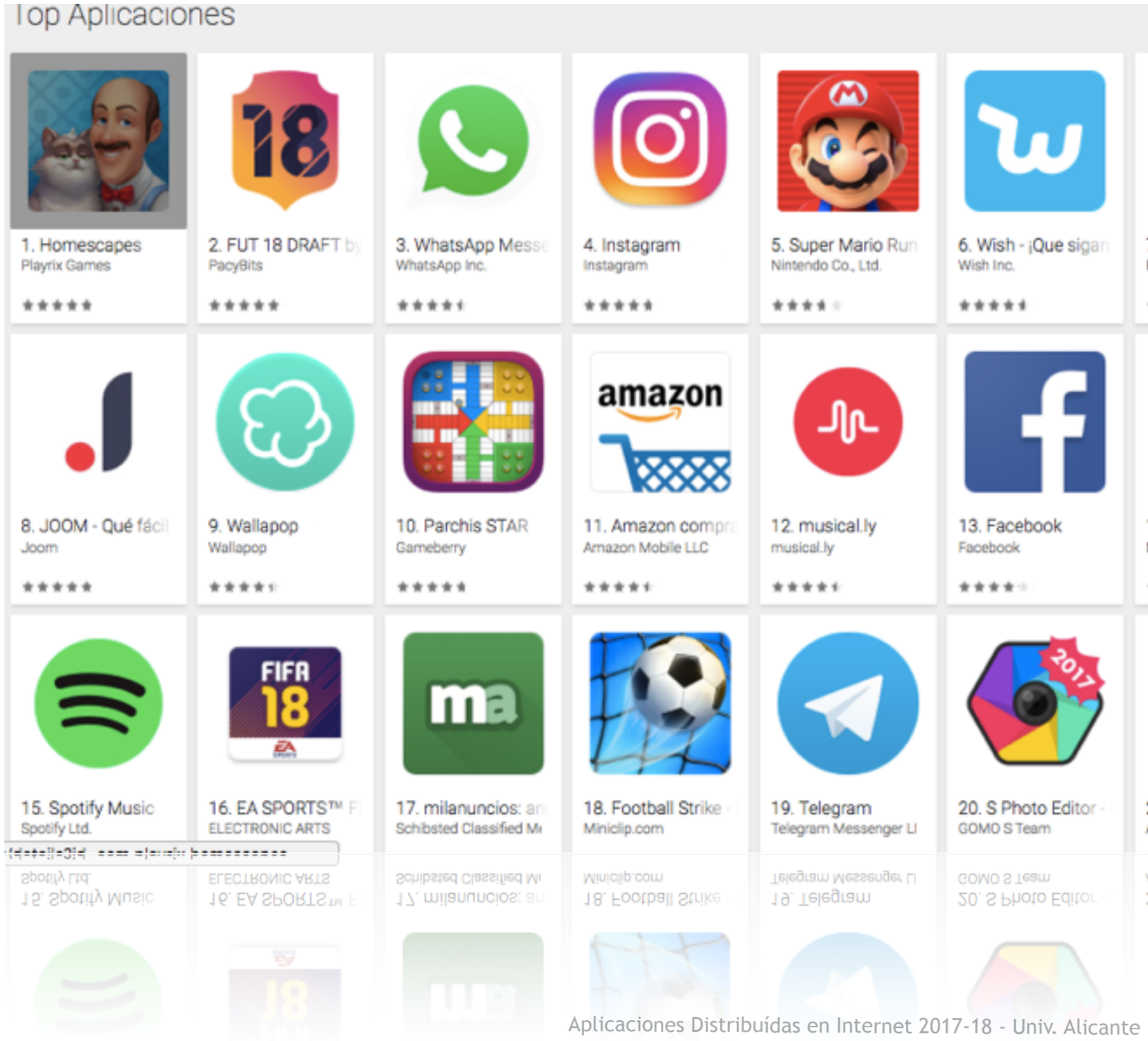
Aún así el backend es complejo

- **Desarrollar** un *backend* y **desplegarlo** en una plataforma PaaS es más sencillo que gestionar nosotros la plataforma, pero **no es una tarea trivial**

- Muchos desarrolladores *frontend* **carecen de experiencia en *backend***

Aún así el backend es complejo

- No solo las apps web necesitan *backend*
- Además los desarrolladores de apps móviles suelen estar todavía más “perdidos” en *backend* que los de *frontend* web



Ejemplo: gestión de usuarios

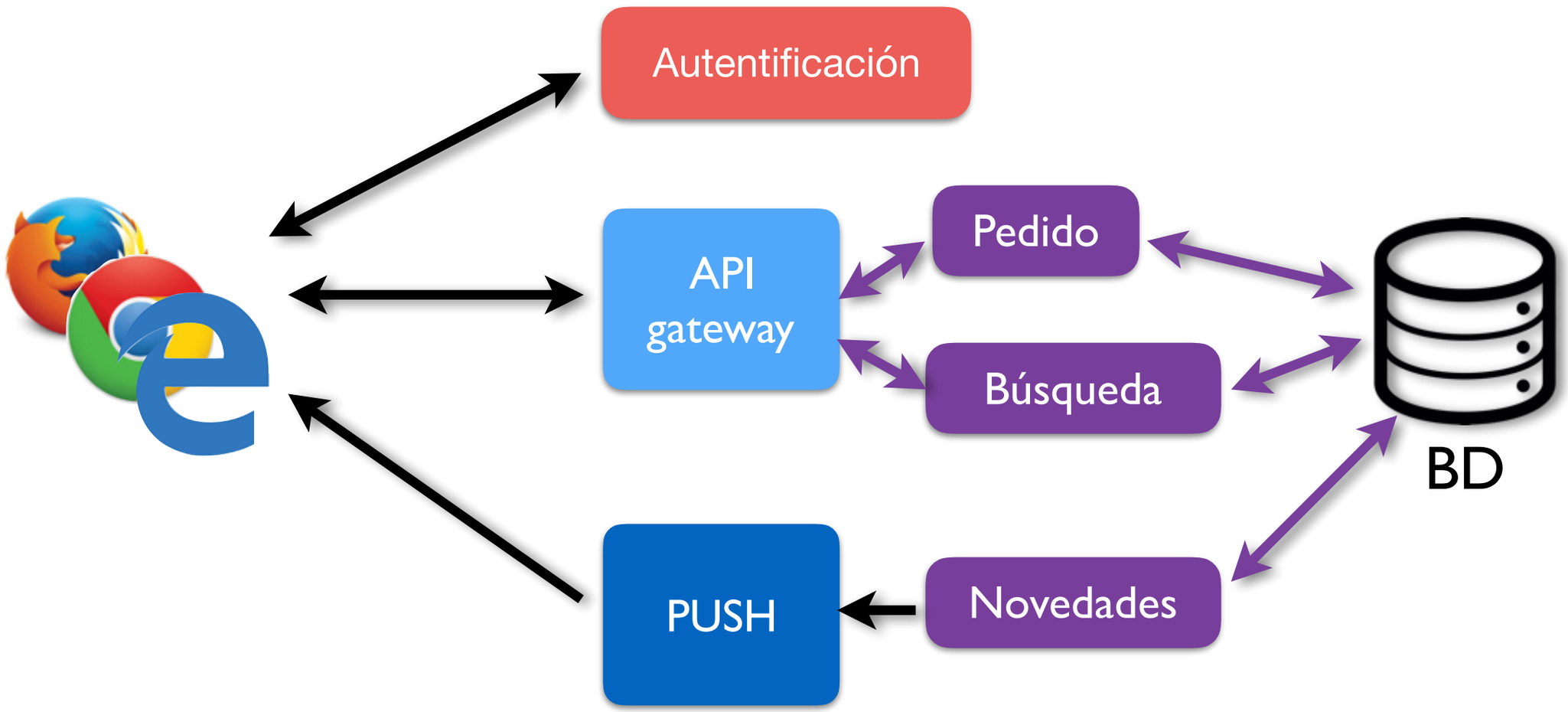
- Gestionar **la BD**
  - Crear y gestionar las tablas: “usuarios”, “roles”,…
  - Cifrar (*hash+salt*) los passwords, no pueden estar en claro en la BD
- Implementar los **casos de uso** asociados a la gestión de usuarios
  - Login, logout
  - CRUD de usuarios
- **Autentificación**:
  - Implementar algún “protocolo” como HTTP Basic o JWT
  - Uso de identidades de redes sociales (“Entra con tu cuenta de Facebook”) - OAuth
- **Pequeños detalles**
  - Enviar email para confirmar registro
  - Implementar reseteo de password por si a alguien se le olvida

Y eso sin contar con que todavía tenemos que implementar el **núcleo de nuestra aplicación**:

- Diseñar el **modelo del dominio**
- Implementar la **lógica de negocio**
- Implementar la **capa de acceso a datos**
- Crear y gestionar la **base de datos** (no el servidor, sino las tablas en sí)

Serverless

- "Externalización" de los **servicios típicos de *backend***
- No significa que no haya **servidor**, sino que es **transparente** para el desarrollador



Aunque no ponemos nada en gris TODO está "externalizado".
Nosotros solo escribimos el código de las partes en ■ y configuramos según necesidades el resto

Tipos de serverless

- **Backend as a Service**: externalización de los servicios típicos de *backend*
  - También llamado **Mobile Backend as a Service** para recalcar la idea de que son servicios muy apropiados para desarrolladores de apps móviles
  - Ejemplos: Kinvey. Firebase, Cloudmine…

kinvey

http://kinvey.com

Firebase

https://firebase.google.com

- **Functions as a Service**: externalización y modularización de la lógica de negocio, que acaba codificándose como un conjunto de funciones independientes (==microservicios)
  - Ejemplos: Amazon Lambda, Azure Functions, …

https://aws.amazon.com/es/lambda

- Últimamente cuando se habla de **serverless** casi siempre se está hablando de **FaaS**

https://azure.microsoft.com/es-es/services/functions/

Tema 2: Aplicaciones web en “la nube”

2.3 (Mobile) Backend as a Service

Servicios típicos de BaaS

Persistencia

Seguridad y autentificación

Integración con terceros

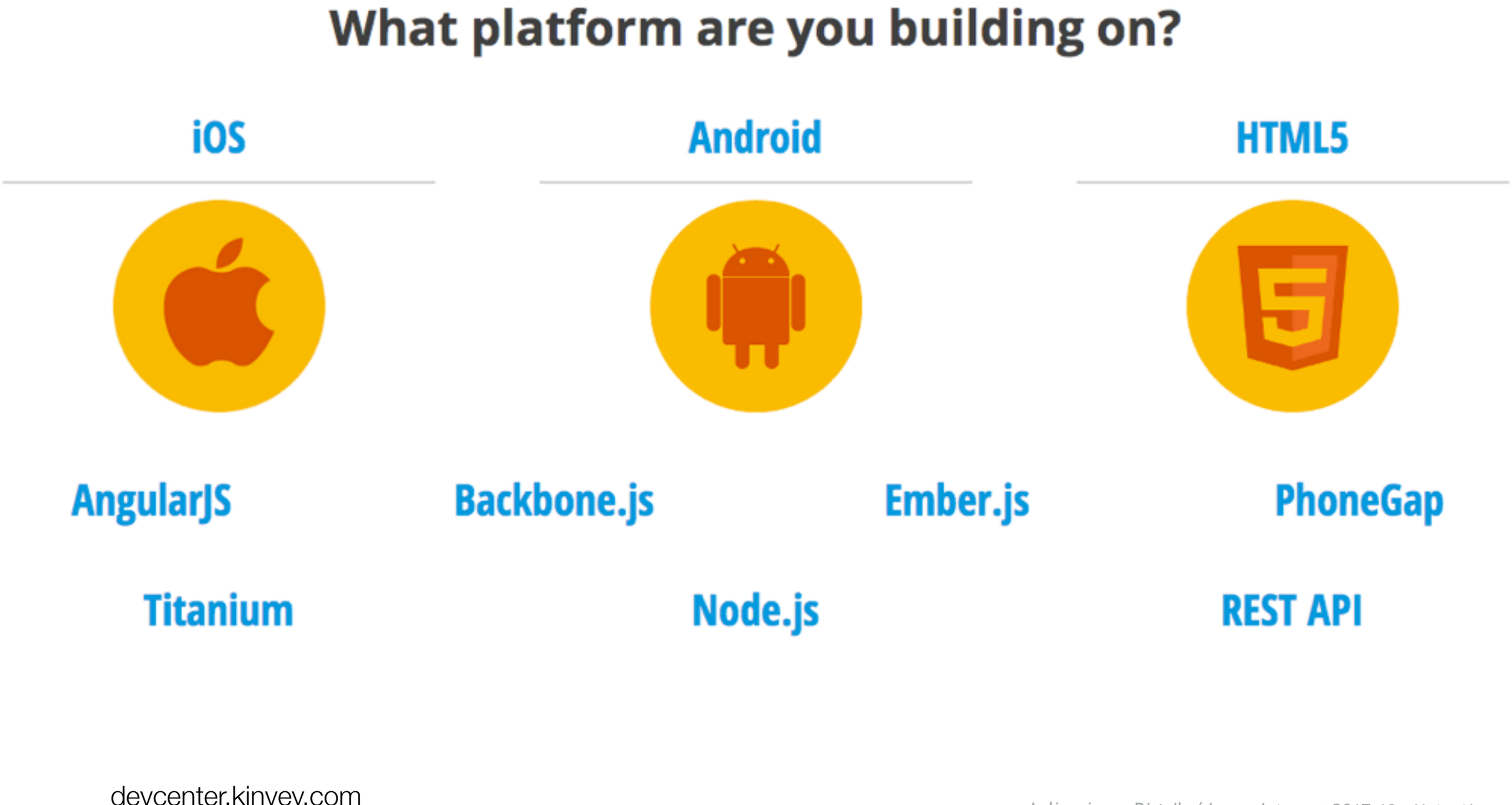
Push

Lógica de negocio en el *backend*

Analíticas, *logs*, ...

Iconos tomados de devcenter.kinvey.com

SDKs multiplataforma



devcenter.kinvey.com

Gracias a los servicios de las plataformas **BaaS**, **desde el cliente** y usando por ejemplo **JavaScript** podemos registrar un nuevo usuario de modo tan sencillo como

```
Kinvey.User.signup({
    username : 'adi',
    password : ‘adi’,
    email : ‘adi@ua.es’
});
```

sin escribir **ni una línea de código en el servidor**

https://ottocol.github.io/ADI1718/teoria/traspas/arquitecturas/demos/demo_signup_kinvey.html

Plataformas BaaS especializadas

Se especializan en un tipo de servicio, aunque muchas empresas comenzaron ofreciendo uno y se han ido expandiendo

- Comunicación en “**tiempo real**”
- Servicios de notificación o ***push***
- **Database** as a service
- **Autentificación**
- **Búsqueda**
- ....

BaaS para “tiempo real”

- Ejemplo: http://pubnub.com: sistema *pub/sub* con baja latencia

PubNub®

```
var PUBNUB_demo = PUBNUB.init({
    publish_key: 'Your Publish Key Here',
    subscribe_key: 'Your Subscribe Key Here'
  });
```

```
PUBNUB_demo.subscribe({
  channel: ‘demo_ADI’,
  message: function(m){console.log(m)}
});
```

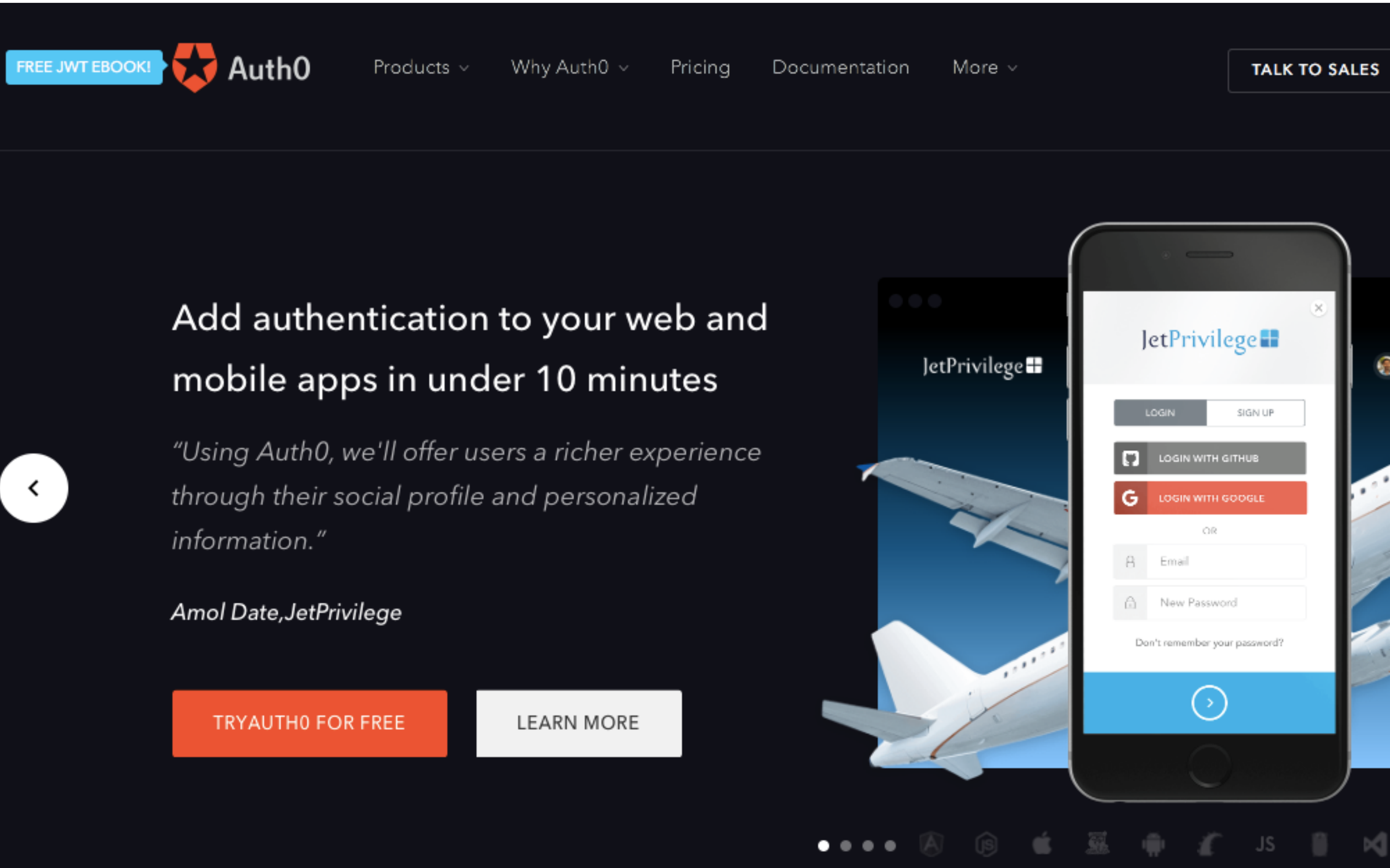
Recepción

```
PUBNUB_demo.publish({
    channel: ‘demo_ADI’,
    message: {"texto":"hola PubNub"}
 });
```

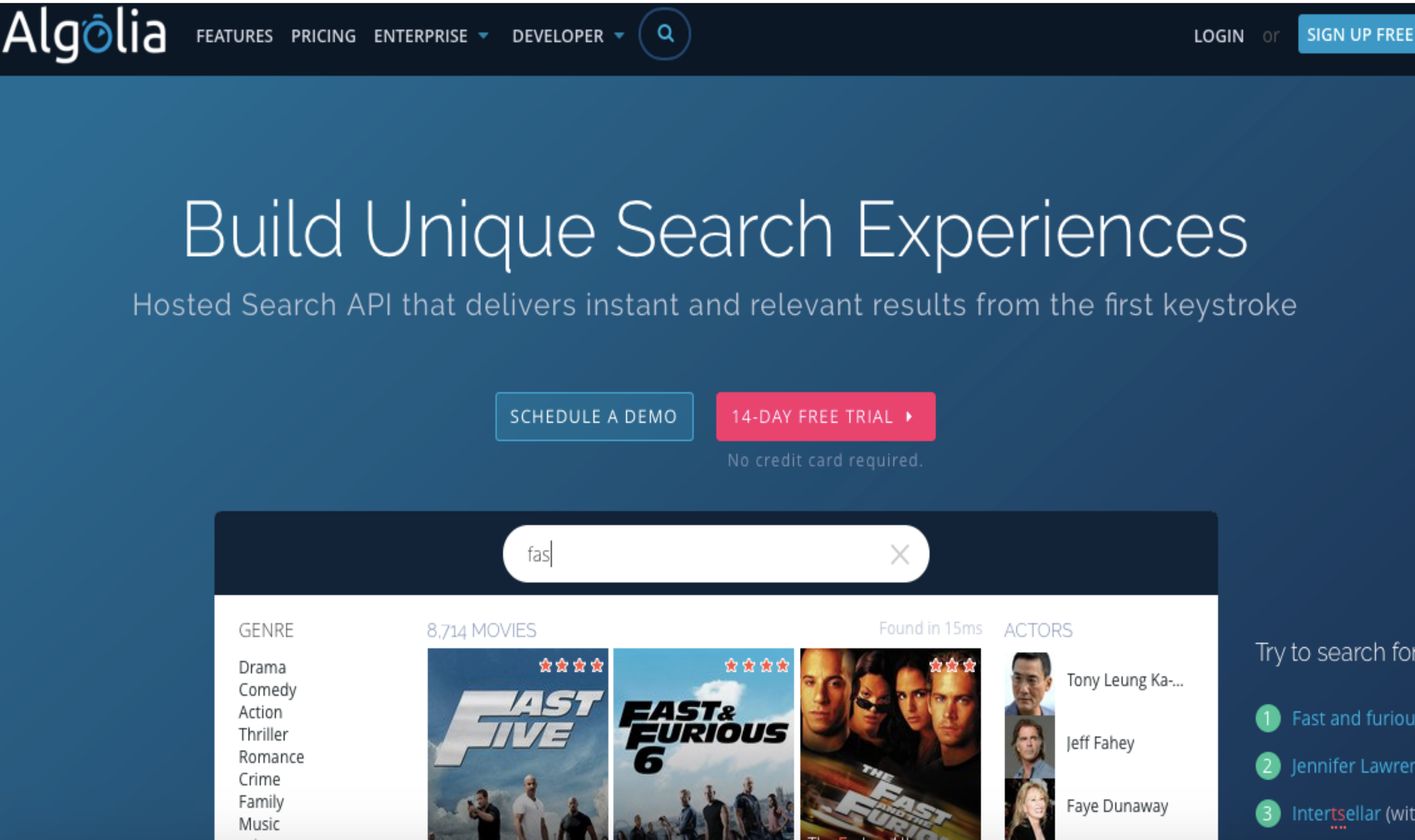
Envío

https://ottocol.github.io/ADI1718/teoria/traspas/arquitecturas/demos/demo_chat_pubnub.html

Autentificación como servicio



Búsqueda como servicio



Bases de datos como servicio

- https://cloudant.com/
- http://www.cloudbase.io/
- https://cloud.google.com/products/cloud-storage
- http://www.dropbox.com/
- http://aws.amazon.com/simpledb/
- http://www.firebase.com/

BaaS “instalable”

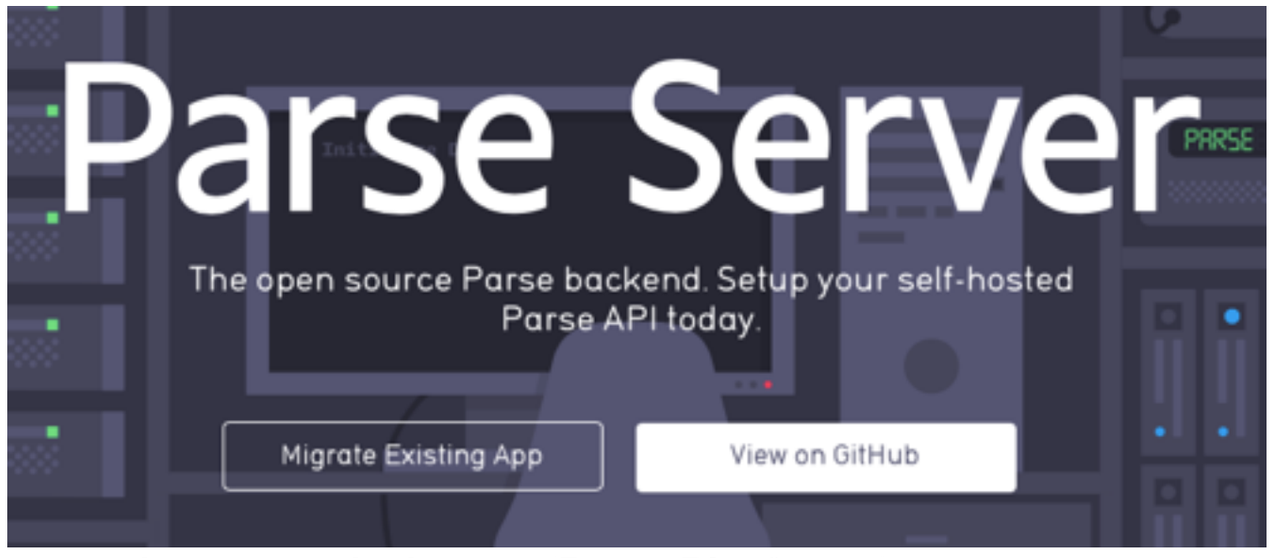
- Plataformas BaaS que podemos **instalar en nuestra propia infraestructura** en lugar de “en la nube”

Hoodie

LiveOak

apache usergrid_

Usergrid



https://github.com/ParsePlatform/parse-server

Otro ejemplo: Firebase

https://firebase.google.com/products/?hl=es-419#develop-features

https://firebase.google.com/docs/?hl=es-419

Tema 6: Aplicaciones web en “la nube”

2.2 Functions as a Service

Características de FaaS

AWS Lambda le permite ejecutar código sin aprovisionar ni administrar servidores. Solo pagará por el tiempo de cómputo que consuma – no se cobra nada cuando el código no se está ejecutando. Con Lambda, puede ejecutar código para casi cualquier tipo de aplicación o servicio back-end – y todo sin administrar nada. Usted solo tiene que cargar el código. Lambda se encargará de todo lo necesario para ejecutar y escalar el código con alta disponibilidad. Puede configurar el código para que se active automáticamente desde otros servicios de AWS o puede llamarlo directamente desde cualquier aplicación web o móvil

https://aws.amazon.com/es/lambda/

Características de FaaS

AWS Lambda le permite ejecutar código sin aprovisionar ni administrar servidores. Solo pagará por el tiempo de cómputo que consuma – no se cobra nada cuando el código no se está ejecutando. Con Lambda, puede ejecutar código para casi cualquier tipo de aplicación o servicio back-end – y todo sin administrar nada. Usted solo tiene que cargar el código. Lambda se encargará de todo lo necesario para ejecutar y escalar el código con alta disponibilidad. Puede configurar el código para que se active automáticamente desde otros servicios de AWS o puede llamarlo directamente desde cualquier aplicación web o móvil

https://aws.amazon.com/es/lambda/

- Serverless

Características de FaaS

AWS Lambda le permite ejecutar código sin aprovisionar ni administrar servidores. Solo pagará por el tiempo de cómputo que consuma – no se cobra nada cuando el código no se está ejecutando. Con Lambda, puede ejecutar código para casi cualquier tipo de aplicación o servicio back-end – y todo sin administrar nada. Usted solo tiene que cargar el código. Lambda se encargará de todo lo necesario para ejecutar y escalar el código con alta disponibilidad. Puede configurar el código para que se active automáticamente desde otros servicios de AWS o puede llamarlo directamente desde cualquier aplicación web o móvil

https://aws.amazon.com/es/lambda/

- Serverless
- No se cobra por tiempo que el servidor está *online*

Características de FaaS

AWS Lambda le permite ejecutar código sin aprovisionar ni administrar servidores. Solo pagará por el tiempo de cómputo que consuma – no se cobra nada cuando el código no se está ejecutando. Con Lambda, puede ejecutar código para casi cualquier tipo de aplicación o servicio back-end – y todo sin administrar nada. Usted solo tiene que cargar el código. Lambda se encargará de todo lo necesario para ejecutar y escalar el código con alta disponibilidad. Puede configurar el código para que se active automáticamente desde otros servicios de AWS o puede llamarlo directamente desde cualquier aplicación web o móvil

https://aws.amazon.com/es/lambda/

- Serverless
- No se cobra por tiempo que el servidor está *online*
- Escalado horizontal automático. Se ejecutará en paralelo el número de copias necesarias

Características de FaaS

AWS Lambda le permite ejecutar código sin aprovisionar ni administrar servidores. Solo pagará por el tiempo de cómputo que consuma – no se cobra nada cuando el código no se está ejecutando. Con Lambda, puede ejecutar código para casi cualquier tipo de aplicación o servicio back-end – y todo sin administrar nada. Usted solo tiene que cargar el código. Lambda se encargará de todo lo necesario para ejecutar y escalar el código con alta disponibilidad. Puede configurar el código para que se active automáticamente desde otros servicios de AWS o puede llamarlo directamente desde cualquier aplicación web o móvil

https://aws.amazon.com/es/lambda/

- Serverless
- No se cobra por tiempo que el servidor está *online*
- Escalado horizontal automático. Se ejecutará en paralelo el número de copias necesarias
- Las funciones se activan con eventos disparados por otros servicios (p.ej. una cola de mensajes, una BD, …) o bien por una petición HTTP

Código de una func. en AWS

- El evento que ha disparado la función es el objeto **event**, y para devolver resultado llamamos a la función **callback**. El objeto **context** da información adicional sobre la propia función

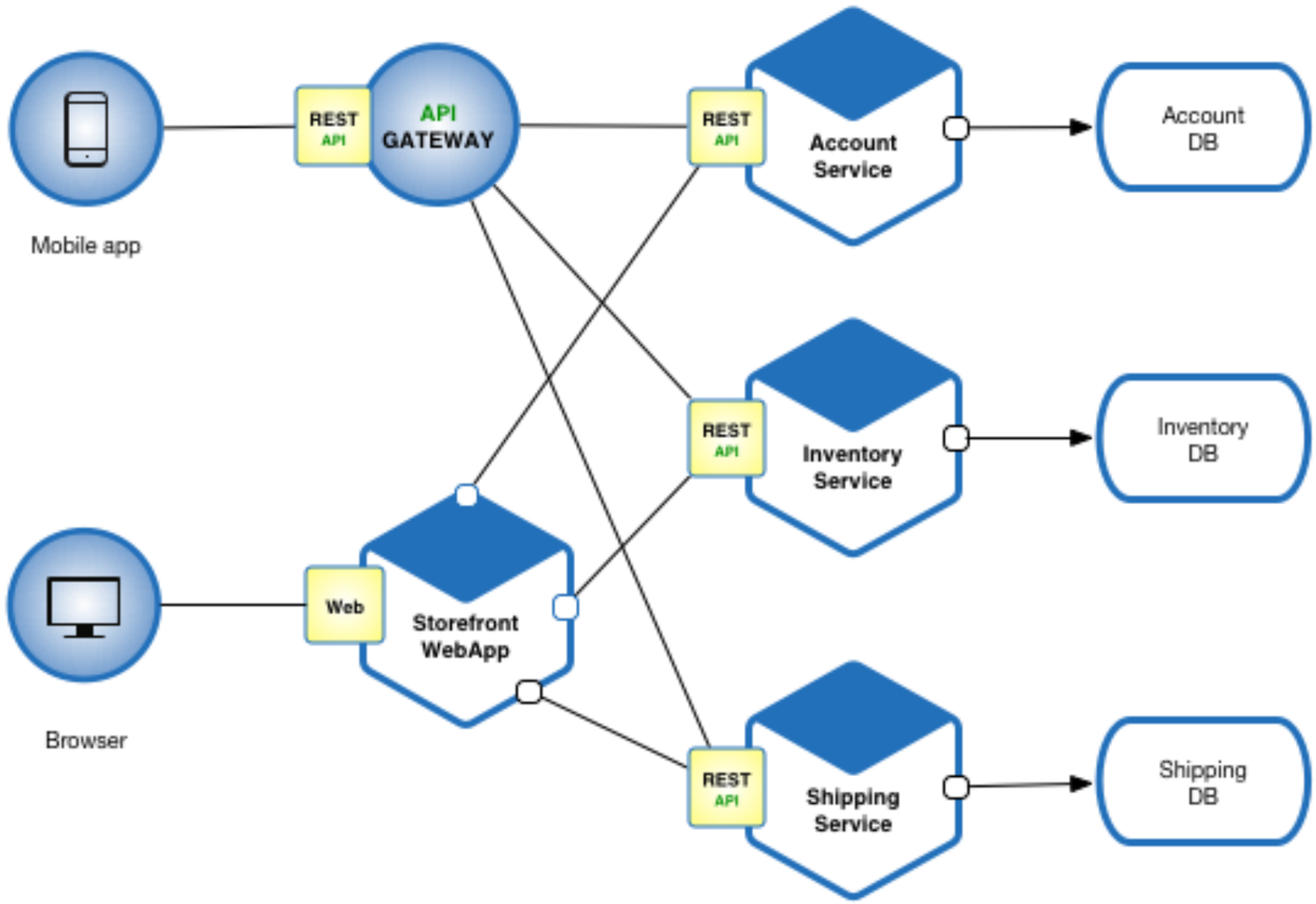
```
console.log('Cargando funcion');

var mensajes = ['Hola', 'Qué tal,', 'Cómo te va,'];

exports.handler = function (event, context, callback) {
    var elegido = Math.floor(Math.random()*mensajes.length)
    var saludo = mensajes[elegido] + ' ' + event.nombre
    callback(null, saludo);
};
```

Microservicios

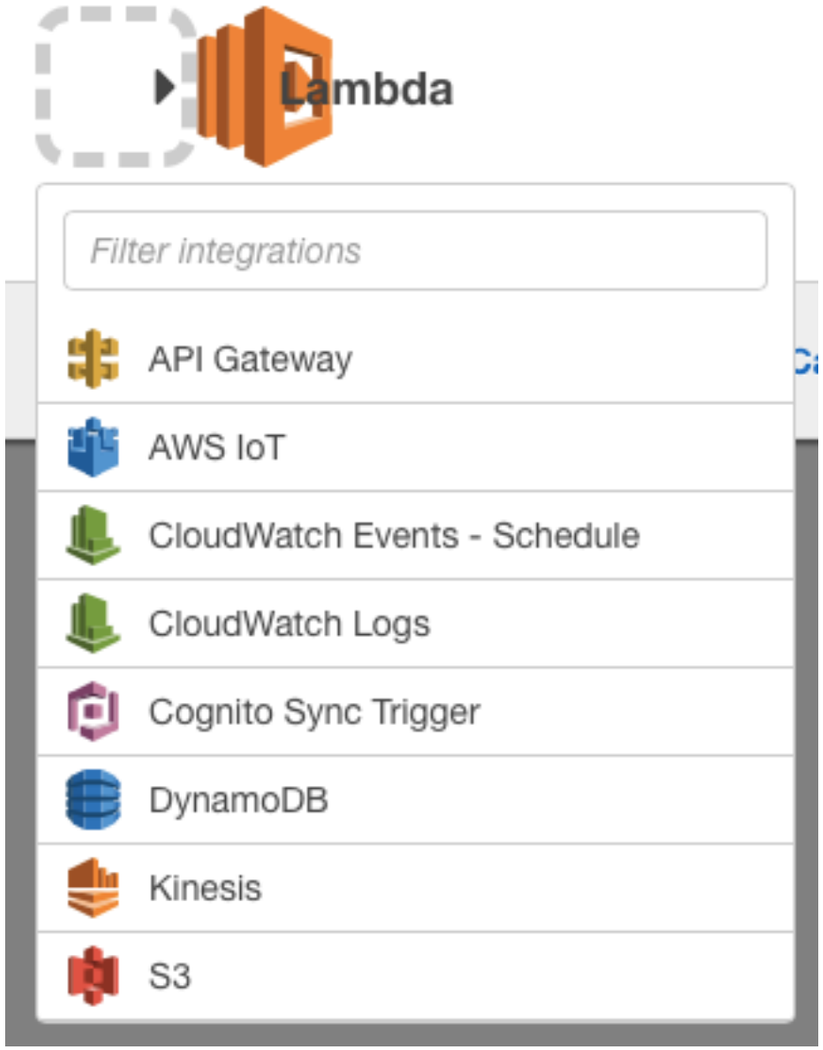
- Arquitectura en la que en lugar de un código monolítico en el servidor tenemos muchos servicios “pequeños”, independientes entre sí y cada uno con su propia BD



Eventos para disparar la función

Las plataformas suelen aceptar peticiones HTTP y también conectan con los servicios propios. Por ejemplo en Amazon:

- Petición HTTP (API Gateway)
- Eventos sobre una BD DynamoBD (p.ej. nuevo registro)
- Eventos sobre un *bucket* de S3 (Simple Storage Service). Por ejemplo, subir un archivo
- Eventos sobre un dispositivo IoT (Internet of Things)
- …



El despliegue es complejo

- Por desgracia actualmente el **despliegue** y **configuración** de FaaS **no es tan sencillo** como debería ser
- Además el proceso es **distinto** según la **plataforma**
- Hay algunos *frameworks* que intentan solucionar estos problemas
  - **Serverless framework**: soporta AWS, Google Cloud, Azure e IBM OpenWhisk
  - **ClaudiaJS**: simplifica el despliegue pero solo para AWS https://claudiajs.com/

https://serverless.com/

https://claudiajs.com/

Limitaciones

- **Recursos** limitados
  - Tiempo de cómputo máximo 5 minutos, se puede fijar menos. Esto impide por ejemplo, usar lambda para trabajos *batch* o en *background*
  - Memoria máxima (por defecto 128Mb)
- **Latencia** en la ejecución
  - Una lambda escrita en Java necesita que arranque la JVM, lo que puede ser un proceso costoso. La plataforma deja “arrancada” la JVM para evitarlo pero si pasa mucho tiempo entre invocación e invocación, libera la memoria. Esto no es tan problemático con Javascript/Python
- **Sin estado**: no podemos asumir que se guarde nada entre ejecución y ejecución. Si queremos guardar algo permanente debemos hacerlo por ejemplo en un servicio de BD

Autoescalado

La plataforma se encarga de **crear automáticamente las copias necesarias** para atender peticiones en paralelo (con un máximo actualmente de 100)

- **Event sources that aren't stream-based** – If you create a Lambda function to process events from event sources that aren't stream-based (for example, Amazon S3 or API Gateway), each published event is a unit of work. Therefore, the number of events (or requests) these event sources publish influences the concurrency.
  You can use the following formula to estimate your concurrent Lambda function invocations:

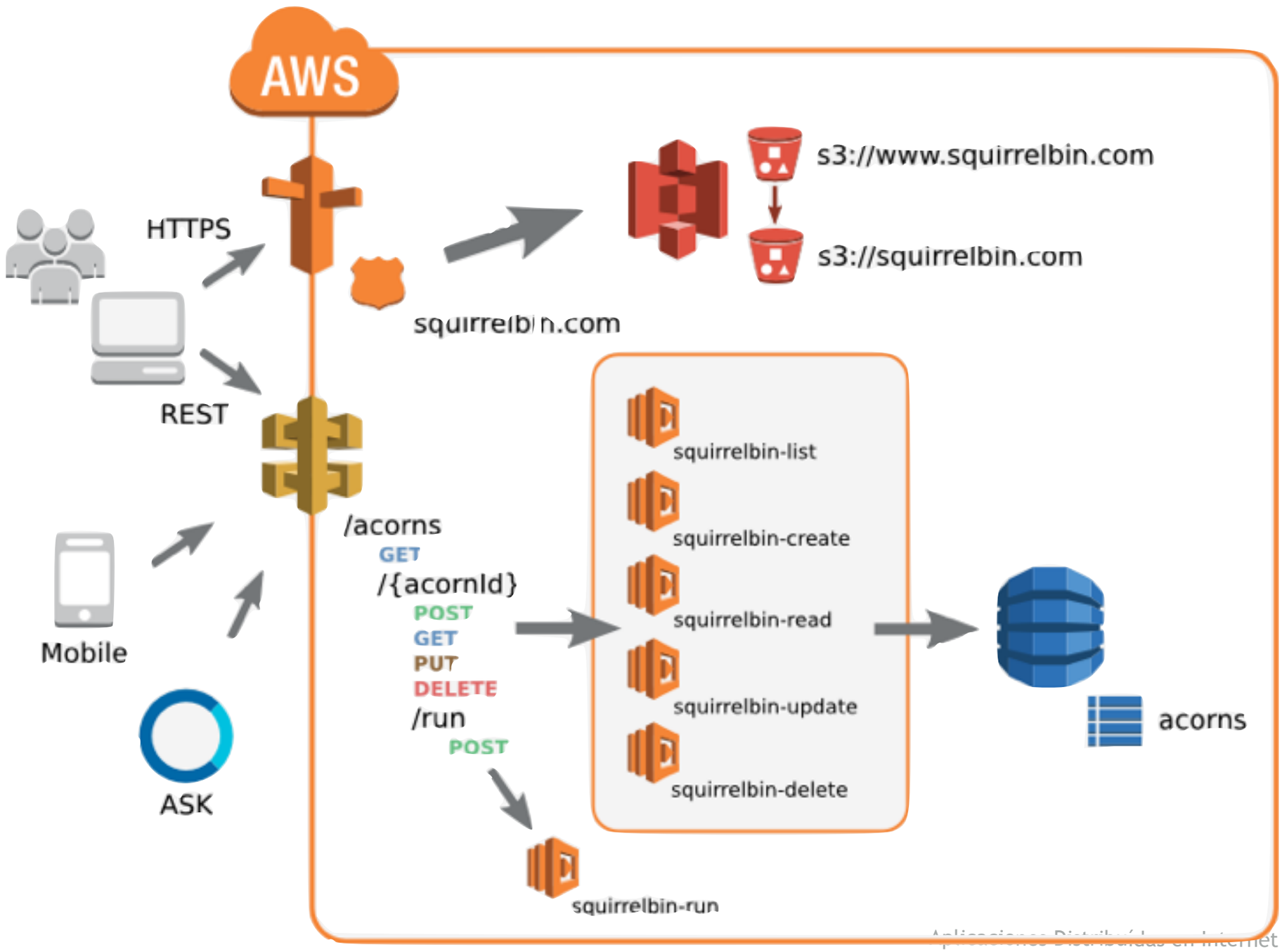
```
events (or requests) per second * function duration
```

  For example, consider a Lambda function that processes Amazon S3 events. Suppose that the Lambda function takes on average three seconds and Amazon S3 publishes 10 events per second. Then, you will have 30 concurrent executions of your Lambda function.

http://docs.aws.amazon.com/lambda/latest/dg/concurrent-executions.html

Tutorial para una app completa

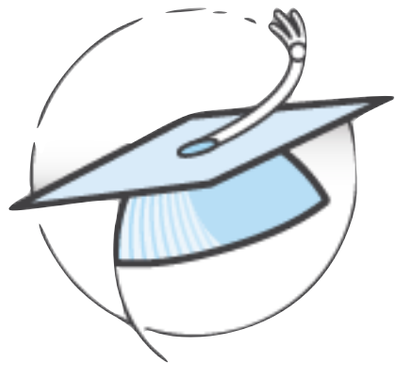
Squirrelbin, una app de ejemplo con lambda+DynamoDB creada por Amazon. Ejemplifica cómo implementar un API REST de tipo CRUD con BD

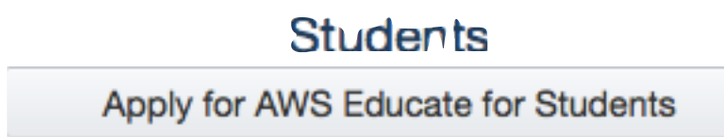


Probar los servicios FaaS

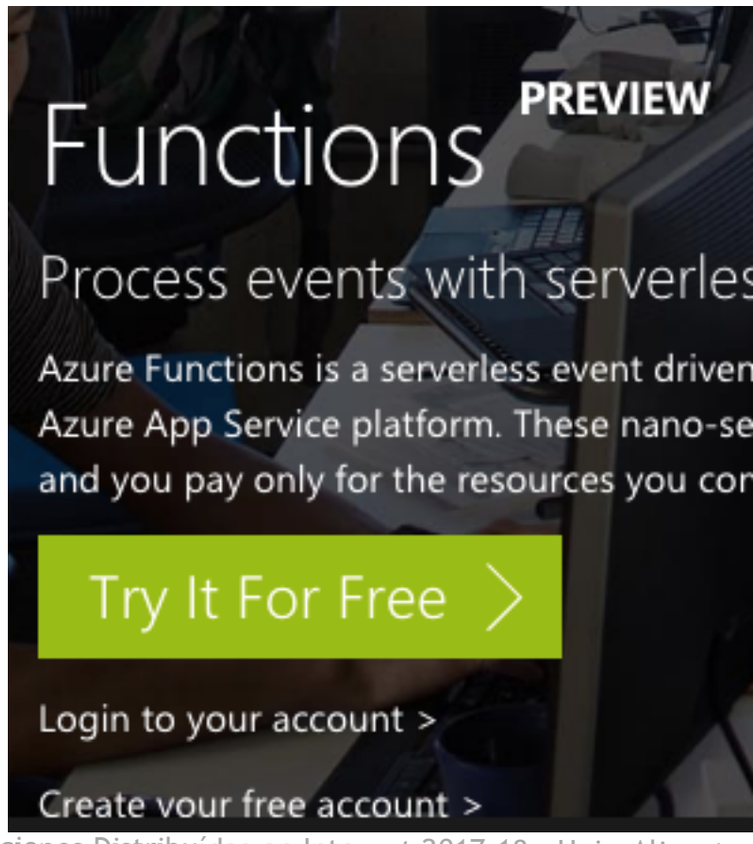
Por desgracia, la mayoría de estos servicios requieren de una tarjeta de crédito para darse de alta, pero hay algunas alternativas

- **Amazon Web Services**: la UA figura desde hace muy poco como centro asociado al programa AWS Educate. Podéis solicitar el alta en https://www.awseducate.com/Application?apptype=student (la cuenta tipo Starter no requiere tarjeta y tiene un crédito de 75$)





- **Azure**: podéis probar las Azure Functions sin tener que daros de alta durante una hora en https://functions.azure.com/try *(se puede repetir las veces que se quiera, pero no se guardan los datos)*



Tema 2: Aplicaciones web en “la nube”

2.4 Conclusiones

Ventajas

- **Simplificación** de la gestión de operaciones
- Escalabilidad
- Reducción de **costes**

Problemas

- **Dependencia** de la plataforma:
  - Prácticamente nula portabilidad
  - …¿qué pasa si desaparece?
- Tecnologías **inmaduras**:
  - Se necesitan **mejores herramientas** para despliegue e integración
  - Todavía no hay un conjunto de **buenas prácticas** aceptadas a la hora de estructurar la aplicación, sobre todo en FaaS

Algunas referencias

- http://martinfowler.com/articles/serverless.html
  - Artículo de un colaborador de Fowler que explica bastante bien y de manera neutral lo que significa *serverless*, sus implicaciones en cuanto a la arquitectura de las aplicaciones y sus ventajas e inconvenientes
- https://github.com/anaibol/awesome-serverless
  - Extensa lista de todo tipo de recursos sobre *serverless*, sobre todo de plataformas que ofrecen estos servicios, pero también de frameworks de desarrollo, libros, artículos introductorios…
- Libro: **Serverless Single Page Apps**, Ben Rady. Pragmatic Programmers, 2016
  - Explica cómo desarrollar una web de tipo “single page app” usando Amazon Web Services