Clase grabada

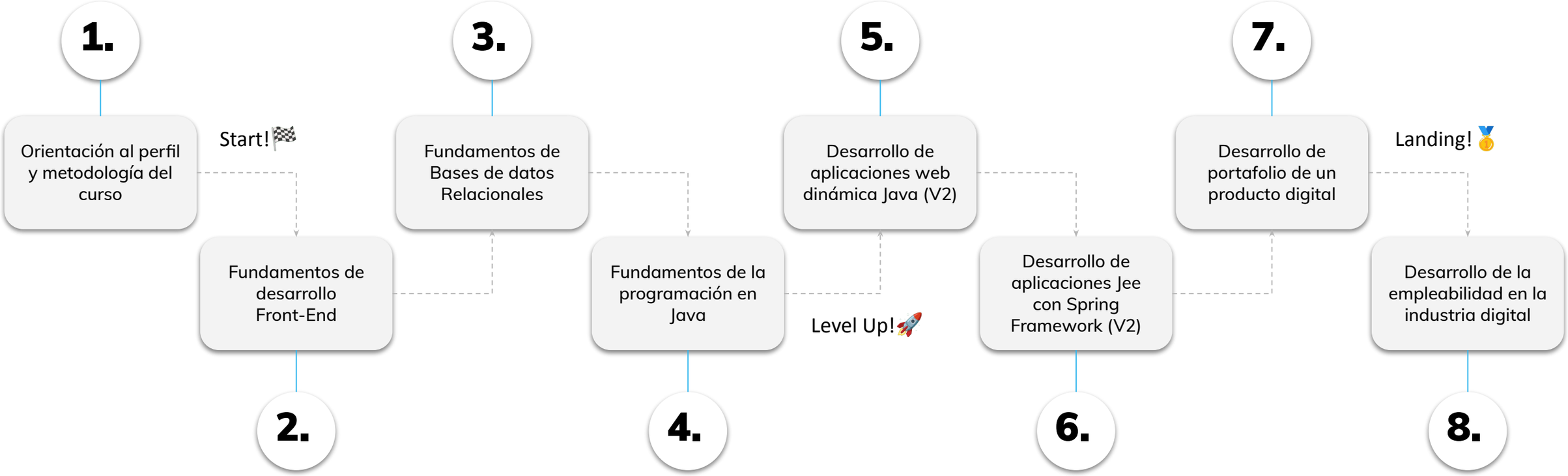
La interoperabilidad entre los sistemas

Plan formativo: Desarrollo de Aplicaciones Full Stack Java Trainee V2.0

English Always | alkemy

HOJA DE RUTA

¿Cuáles **skill** conforman el programa?



English Always | alkemy

REPASO CLASE ANTERIOR

En la clase anterior trabajamos 📚:

- ✓ *El Estado Representacional de Transferencia (REST)*

English Always | alkemy

LEARNING PATHWAY

N° de la unidad . ¿Sobre qué temas trabajaremos?

6.5 Start!🏁

La interoperabilidad entre los sistemas

JSON

JSON

La notación JSON para el traspaso de información

English Always | alkemy

OBJETIVOS DE APRENDIZAJE

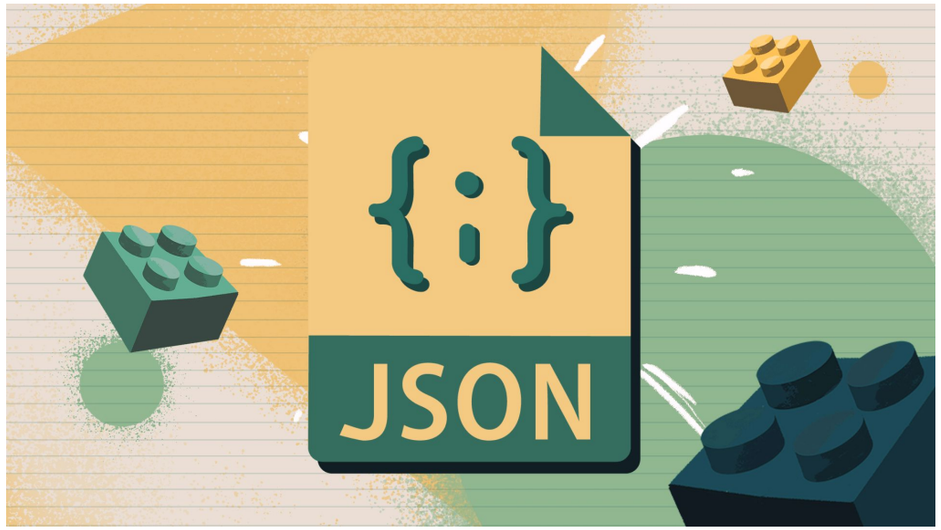
¿Qué aprenderemos?

- ***Comprender la notación JSON y sus características***

JSON

JSON

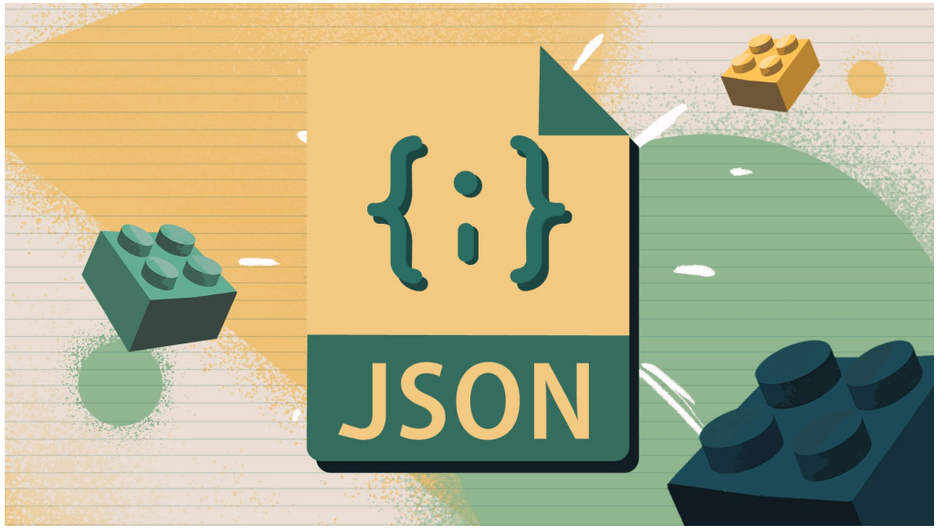
JSON, que significa "**J**ava**S**cript **O**bject **N**otation", es un formato de intercambio de datos ligero y legible por humanos. Fue derivado originalmente del lenguaje de programación JavaScript, pero se ha convertido en un formato independiente que es ampliamente utilizado en el intercambio de datos en la web y en aplicaciones.



JSON

La notación JSON se caracteriza por utilizar una estructura de pares **clave-valor**. Los datos se organizan en objetos, arrays, números, cadenas, booleanos y valores nulos. La simplicidad de su estructura permite la fácil interpretación por parte de humanos y máquinas.

Se basa en dos estructuras de datos principales: **objetos** y **arrays**.



JSON

Un **objeto** en JSON es una colección **no ordenada** de pares clave-valor. Cada clave debe ser una cadena de caracteres y debe ser única dentro del objeto. Los valores pueden ser strings, números, booleanos, null, objetos o arrays.

```
{
    "nombre": "Juan",
    "edad": 30,
    "casado": false,
    "hijos": null,
    "direcciones": {
        "casa": "123 Calle Principal",
        "trabajo": "456 Calle Secundaria"
    },
    "hobbies": ["lectura", "cine", "viajes"]
}
```

JSON

Un **array** es una secuencia ordenada de valores. Los valores pueden ser de cualquier tipo de datos JSON, incluyendo objetos y otros arrays.

```
["manzana", "naranja", "banana", {"fruta": "uva"}, [1, 2, 3]]
```

JSON

Algunas **diferencias** notables en contraste **con XML** son:

- **Sintaxis**: JSON tiene una sintaxis **más concisa y fácil** de leer en comparación con XML. Las etiquetas en XML pueden hacer que los documentos sean más verbosos.
- **Peso del Documento**: JSON **tiende a ser más ligero** porque tiene menos caracteres de marcado que XML, lo que resulta en una transmisión de datos más eficiente.
- **Amplia Adopción en la Web**: JSON es el formato de datos nativo para JavaScript, lo que facilita su manipulación en aplicaciones web. Además, es ampliamente aceptado en muchas otras plataformas y lenguajes de programación.
- **Facilidad de Mapeo a Estructuras de Datos en Programación**: La estructura de JSON se mapea fácilmente a estructuras de datos en muchos lenguajes de programación, simplificando la integración de datos en aplicaciones.

JSON y Spring

JSON y Spring

La utilización de JSON en entornos de desarrollo Java, especialmente en el marco de trabajo Spring, ofrece diversas **ventajas** que van más allá de simplemente facilitar el intercambio de datos. Las principales son:

1. **Ligereza y Eficiencia**

2. **Interoperabilidad y Flexibilidad**

3. **Facilidad de Lectura y Depuración**

JSON y Spring

Java es conocido por ser un lenguaje de programación robusto y orientado a objetos, pero en entornos de desarrollo modernos, la eficiencia en el intercambio de datos es esencial. Aquí es donde JSON brilla, y una de las principales ventajas radica en su ligereza.

- **Estructura de Datos Compacta y Fácil de Procesar**: JSON utiliza una estructura de datos simple que es fácil de entender y procesar tanto para las aplicaciones como para los desarrolladores. En comparación con otros formatos de intercambio de datos más verbosos, como XML, JSON tiende a ser más compacto, lo que resulta en una transmisión de datos más eficiente.

JSON y Spring

- **Deserialización Eficiente en Java con Bibliotecas JSON**: JSON se integra naturalmente con bibliotecas de deserialización en Java, como Jackson y Gson, que permiten convertir datos JSON en objetos Java de manera rápida y eficiente.

```
// Ejemplo de deserialización con Jackson en Spring
String jsonString = "{\"nombre\": \"Juan\", \"edad\": 30, \"casado\": false}";
ObjectMapper objectMapper = new ObjectMapper();
Persona persona = objectMapper.readValue(jsonString, Persona.class);
```

En este ejemplo, la clase Persona puede tener campos que coincidan directamente con las claves en el objeto JSON, simplificando el proceso de deserialización.

JSON y Spring

- **Menor Consumo de Ancho de Banda y Recursos**: La ligereza de JSON se traduce en un menor consumo de ancho de banda al transmitir datos a través de la red. Esto es especialmente beneficioso en entornos donde la eficiencia de recursos es crucial, como en aplicaciones móviles o en sistemas distribuidos.

- **Rendimiento Optimizado en Aplicaciones Spring**: La facilidad con la que los datos JSON pueden ser convertidos a objetos Java facilita la manipulación de datos, mejorando la eficiencia y la capacidad de respuesta de las aplicaciones.

JSON y Spring

- **Compatibilidad con Diversos Lenguajes y Plataformas**: Una de las ventajas fundamentales de JSON es su capacidad para ofrecer interoperabilidad entre diferentes lenguajes y plataformas. Esto se alinea perfectamente con la naturaleza políglota de las aplicaciones modernas, donde diversas tecnologías colaboran para crear sistemas complejos.

- **Integración Natural con Tecnologías Web**: En el contexto de Java y Spring, la integración de JSON es especialmente valiosa en aplicaciones web. La mayoría de los navegadores web y clientes HTTP admiten nativamente la serialización y deserialización de datos JSON, lo que facilita la comunicación entre el frontend y el backend de una aplicación.

JSON y Spring

- **Flexibilidad en la Representación de Datos**: Puedes modelar datos complejos, anidando objetos y arrays, lo que facilita la representación de relaciones y estructuras de datos más avanzadas.

- **Integración con Servicios Web y APIs**: En aplicaciones Spring que interactúan con servicios web y APIs, JSON se ha convertido en el formato dominante para el intercambio de datos. La mayoría de los servicios web modernos ofrecen endpoints que aceptan y devuelven datos en formato JSON, lo que simplifica la integración de diferentes componentes de software.

JSON y Spring

- **Legibilidad para Desarrolladores y Máquinas**: Una de las características distintivas de JSON es su legibilidad tanto para los desarrolladores como para las máquinas. La sintaxis simple y clara facilita la comprensión de la estructura de los datos, lo que es esencial durante el desarrollo y la depuración de aplicaciones.

- **Herramientas de Visualización y Depuración Integradas**: En entornos de desarrollo Java, como Eclipse o IntelliJ, las herramientas de visualización y depuración están optimizadas para trabajar con datos JSON. Los desarrolladores pueden inspeccionar y visualizar fácilmente los objetos Java generados a partir de datos JSON durante la ejecución de la aplicación.

JSON y Spring

- **Facilidad en la Creación de Datos de Prueba:** Durante el desarrollo y las pruebas, la capacidad de representar datos de manera legible y fácil con JSON facilita la creación de datos de prueba realistas. Los desarrolladores pueden crear fácilmente conjuntos de datos complejos y variados para probar diferentes escenarios de aplicación.

La elección de JSON en el desarrollo de aplicaciones Spring y Java no solo es una cuestión de conveniencia, sino que también contribuye a la eficiencia, la legibilidad del código y la capacidad de respuesta de las aplicaciones. La capacidad de representar datos de manera clara y eficiente es esencial en el panorama actual del desarrollo de software.

JSON:Casos de Uso

JSON:Casos de Uso

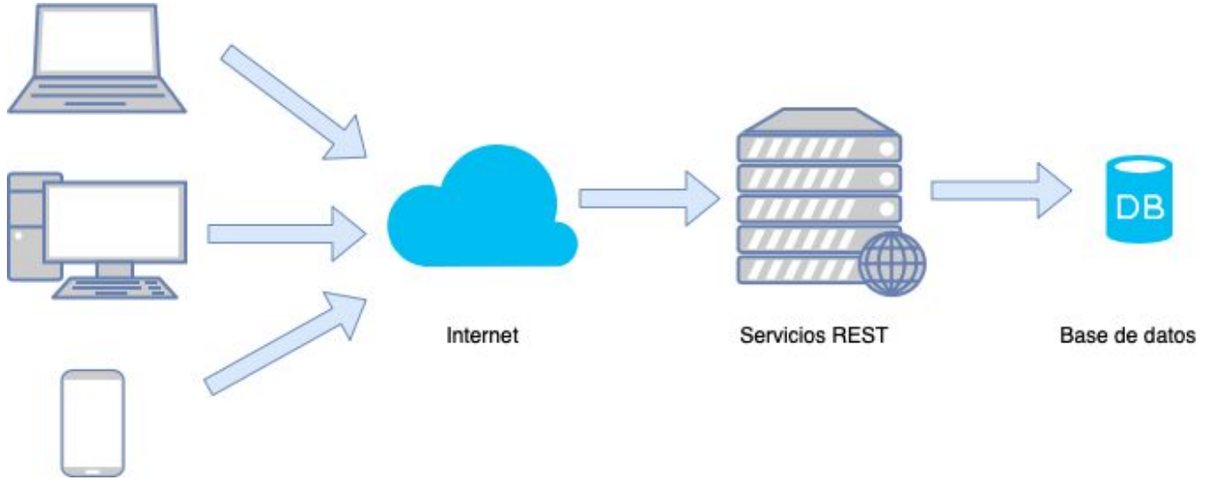
La adopción generalizada de JSON en el desarrollo de aplicaciones Spring y Java no solo se basa en sus ventajas teóricas, sino también en su aplicabilidad práctica en una variedad de escenarios del mundo real.

Dos típicos casos de uso en los cuales JSON desempeña un papel crucial son: **Servicios Web RESTful** y **Aplicaciones de Una Página** (SPA).

JSON:Casos de Uso

1- Servicios Web RESTful

Los servicios web RESTful (Representational State Transfer) se han convertido en el estándar predominante para el desarrollo de APIs en la arquitectura web. Estos servicios utilizan los principios fundamentales de REST para facilitar la comunicación entre sistemas distribuidos a través de HTTP. JSON, con su estructura de datos liviana y fácilmente serializable, se integra perfectamente en la arquitectura RESTful.



JSON:Casos de Uso

Intercambio de Datos en Formato JSON en Servicios RESTful

En un servicio web RESTful, la representación de recursos se realiza comúnmente en formato JSON. Los recursos, que pueden ser objetos complejos o colecciones de datos, se transmiten entre el cliente y el servidor en el cuerpo de las solicitudes y respuestas HTTP. La simplicidad y la legibilidad de JSON hacen que sea fácil para los desarrolladores entender y manipular estos datos en el proceso de desarrollo y depuración.

JSON:Casos de Uso

Intercambio de Datos en Formato JSON en Servicios RESTful

En un servicio web RESTful, la representación de recursos se realiza comúnmente en formato JSON. Los recursos, que pueden ser objetos complejos o colecciones de datos, se transmiten entre el cliente y el servidor en el cuerpo de las solicitudes y respuestas HTTP. La simplicidad y la legibilidad de JSON hacen que sea fácil para los desarrolladores entender y manipular estos datos en el proceso de desarrollo y depuración.

```
// Ejemplo de respuesta JSON en un servicio RESTful
{
    "id": 1,
    "titulo": "Artículo de Ejemplo",
    "contenido": "Este es un artículo de ejemplo sobre el uso de JSON
}
```

JSON:Casos de Uso

Deserialización de Datos en el Servidor Spring

En el lado del servidor, en un entorno Spring, la deserialización de datos JSON se simplifica mediante bibliotecas como Jackson. Los controladores de Spring pueden recibir datos JSON en el cuerpo de las solicitudes y automáticamente deserializarlos en objetos Java, facilitando el procesamiento de estos datos en la lógica del servidor.

```java
// Controlador Spring que maneja una solicitud POST con datos JSON
@RestController
@RequestMapping("/articulos")
public class ArticuloController {

    @PostMapping
    public ResponseEntity<String> crearArticulo(@RequestBody Articulo nuevoArticulo) {
        // Lógica para crear un nuevo artículo
        return ResponseEntity.ok("Artículo creado con éxito");
    }
}
```

JSON:Casos de Uso

Implementación de Versionamiento de API con JSON

En servicios web RESTful, la evolución de la API es un factor crítico. JSON facilita la implementación de versionamiento de API, donde nuevas versiones de los recursos pueden introducirse sin romper la compatibilidad con las versiones anteriores. Los clientes pueden adaptarse a cambios incrementales en la estructura de los datos JSON sin interrupciones significativas.

JSON:Casos de Uso

2- El Auge de las Aplicaciones de Una Página (SPA)

Las Aplicaciones de Una Página (SPA) se han vuelto populares en el desarrollo web moderno debido a su capacidad para proporcionar experiencias de usuario fluidas y rápidas al cargar solo una página HTML y actualizar dinámicamente el contenido. JSON desempeña un papel fundamental en estas aplicaciones al facilitar la transferencia eficiente de datos entre el cliente y el servidor.

JSON:Casos de Uso

Manejo de Navegación y Estado con JSON

En una SPA, la navegación entre diferentes vistas y el manejo del estado de la aplicación se gestionan de manera dinámica. JSON se utiliza para representar y transmitir información sobre el estado de la aplicación, lo que facilita el manejo del enrutamiento y la persistencia del estado del cliente.

JSON:Casos de Uso

Validación de Datos JSON en Spring

La validación de datos es un aspecto crucial en cualquier aplicación, y Spring proporciona mecanismos sólidos para validar datos JSON. Utilizando anotaciones como @Valid y @RequestBody, es posible validar automáticamente los datos JSON recibidos en los controladores de Spring.

```
@PostMapping
public ResponseEntity<String> crearUsuario(@Valid @RequestBody Usuario nuevoUsuario) {
    // Lógica para crear un nuevo usuario
    return ResponseEntity.ok("Usuario creado con éxito");
}
```

En este ejemplo, la anotación @Valid activa la validación del objeto Usuario según las reglas definidas en su clase.

JSON:Casos de Uso

Integración con Spring Data y Bases de Datos NoSQL

Spring Data proporciona soporte para la integración con bases de datos NoSQL, y JSON se adapta naturalmente a este entorno. Bases de datos NoSQL como MongoDB almacenan datos en formato BSON (Binary JSON), lo que facilita la integración directa con objetos JSON en el código de la aplicación.

```
// Ejemplo de entidad MongoDB utilizando Spring Dat
@Document(collection = "usuarios")
public class Usuario {

    @Id
```

En este ejemplo, la entidad Usuario se almacena en una colección MongoDB, y los campos de la entidad se mapean directamente a campos JSON en la base de datos

Momento:

Time-out!

⏰15 min.

Ejercicio
JSON

JSON

Contexto: 🙌

Vamos a repasar los conceptos más importantes respondiendo algunas preguntas frecuentes sobre el formato JSON.

Consigna: ✍️ Respondan levantando la mano!

1- ¿Qué significa la abreviatura JSON y cuál es su propósito principal en el desarrollo de software?

2- ¿Qué es la deserialización y por qué es importante en el contexto de JSON?

3- Describe un caso de uso práctico donde la notación JSON sería preferible sobre otros formatos de intercambio de datos.

Tiempo🕐: *15 minutos*

¿Alguna consulta?

RESUMEN

¿Qué logramos en esta clase?

- ✓ ***Comprender el concepto y las características de JSON***

English Always | alkemy

#WorkingTime

Continuemos ejercitando

¡Antes de cerrar la clase! Te invitamos a: 👇👇👇

1. Repasar nuevamente la grabación de esta clase
2. Revisar el material compartido en la plataforma de Moodle (lo que se vio en clase y algún ejercicio adicional)
   a. *Lectura Modulo 6, Lección 4: página 4*
3. Traer al próximo encuentro, todas tus dudas y consultas para verlas antes de iniciar nuevo tema.

¡Muchas Gracias!

Nos vemos en la próxima clase 👋🏻

English Always | alkemy

Momento:

Time-out!

⏰15 min.