Aufgabe 1: Hello World Endpoint ⭐

Erstelle eine einfache Spring-Boot-Anwendung, die einen "Hello, World!"-String über einen REST-Endpunkt bereitstellt.

Anforderungen:

1. Erstelle ein neues Spring-Boot-Projekt mit `Spring Web`.
2. Implementiere einen Controller mit einem Endpunkt `/`.
3. Rückgabe: Der Text "Hello, World!".

Test:

- Starte die Anwendung und prüfe den Endpunkt unter `http://localhost:8080`.

Aufgabe 2: Greet-Me Applikation ⭐⭐

Erstelle eine Anwendung, die den Benutzer mit einer zufälligen Begrüßung in verschiedenen Sprachen begrüßt.

Anforderungen:

1. Erstelle einen REST-Endpunkt `/greet`.
2. Rückgabe: Eine Begrüßung in einer zufälligen Sprache mit einem festen Namen, z. B.: "Hello, Alex!".
3. Beispielsprachen: Englisch, Deutsch, Französisch, Spanisch.
4. Implementiere die Auswahl der Sprache mit einer zufälligen Funktion.

Beispielausgabe:

- `Hola, Alex!`
- `Bonjour, Alex!`

Test:

- Starte die Anwendung und rufe den Endpunkt mehrmals auf, um die verschiedenen Begrüßungen zu prüfen.

Aufgabe 3: Eigener Port und Benutzerdefinierte Konfiguration ⭐⭐

Passe die Standardkonfiguration einer Spring-Boot-Anwendung an.

Anforderungen:

1. Ändere den Serverport auf `9090`.
2. Füge in `application.properties` eine neue Property `app.name=MySpringApp` hinzu.
3. Implementiere einen Controller, der die Anwendungskonfiguration liest und unter `/config` ausgibt.

Beispielausgabe:

```
{
  "port": 9090,
  "name": "MySpringApp"
}
```

Test:

- Starte die Anwendung und prüfe den Endpunkt unter `http://localhost:9090/config`.

Aufgabe 4: Erweiterte Greet-Me Applikation mit Namen ⭐⭐⭐

Erweitere die "Greet-Me" Applikation, sodass sie zusätzlich den Namen des Benutzers berücksichtigt.

Anforderungen:

1. Erstelle einen REST-Endpunkt `/greet/{name}`.
2. Rückgabe: Eine Begrüßung in einer zufälligen Sprache mit dem angegebenen Namen.
3. Beispiel: `/greet/Lisa` könnte "Hallo, Lisa!" oder "Salut, Lisa!" zurückgeben.

Test:

- Rufe den Endpunkt mit verschiedenen Namen auf und prüfe die Antworten.

Aufgabe 5: Integration mit H2-Datenbank ⭐⭐⭐

Erstelle eine Spring-Boot-Anwendung, die eine einfache Datenbankintegration bietet.

Anforderungen:

1. Füge `Spring Data JPA` und `H2 Database` als Abhängigkeiten hinzu.
2. Konfiguriere die H2-Datenbank in `application.properties`.
3. Erstelle eine JPA-Entity `User` mit Feldern `id`, `name` und `email`.
4. Implementiere ein Repository und einen Endpunkt `/users`, der eine Liste aller Benutzer zurückgibt.

Beispielausgabe:

```
[
  {"id": 1, "name": "John Doe", "email": "john.doe@example.com"},
  {"id": 2, "name": "Jane Smith", "email": "jane.smith@example.com"}
]
```

Test:

- Starte die Anwendung und rufe den Endpunkt `/users` auf.