Backend Developer Challenge

El propósito del challenge es testear tus habilidades para implementar una RESTFull API. En este escenario simplificado, la unidad de negocios Mercado Crédito quiere emitir fácilmente nuevos préstamos a los usuarios y averiguar cuál es el valor y el volumen de la deuda pendiente.

Cabe mencionar que existen 3 targets diferentes de usuarios, dependiendo de la cantidad de préstamos tomados y el monto de los mismos en el último año. Además, cada target tiene una tasa de interés propia y un tope máximo a solicitar por préstamo.

- NEW:
    - 0 < cant < 2
    - amount_total < 100000
    - rate: 0.15
    - max: 500000
- FREQUENT:
    - 2 <= cant < 5
    - 100000 < amount_total < 500000
    - rate: 0.10
    - max: 1000000
- PREMIUM:
    - cant > 5
    - amount_total > 500000
    - rate: 0.05
    - max: 5000000

donde
amount_total: es el volumen de préstamo tomado en el último año
cant: es la cantidad de préstamos pedidos en el último año.
rate: tasa de interés (en formato decimal).
max: Cantidad máxima a solicitar por préstamo.

Cabe aclarar que estas condiciones pueden variar según los balances del negocio, pudiendo cambiar los topes máximos, la cantidad, la tasa de interés o el monto total a considerar en cada categoría, por lo cual, a la hora de modelar la solución, esto debería ser lo suficientemente versátil.

La administración necesita realizar un seguimiento de la cantidad de dinero prestada y los pagos realizados, por usuario, por target de usuario y a nivel general.

Consideraciones: Los préstamos se pagan en cuotas mensuales.

Endpoints:

1. ***Solicitud de préstamo***:

Crea una solicitud de préstamo. Los préstamos se aceptan y guardan automáticamente en una base de datos.

Payload:

- amount: monto del préstamo.
- term: número de cuotas.
- user_id: identificador de usuario.

Example:

```
{
    "amount": 1000,
    "term": 12,
    "user_id": 12
}
```

Response:
loan_id: identificador único.
installment: cuota mensual del préstamo.

Notes:
Fórmula de pago de préstamos:
r = rate / 12.
Cuota (mensual) = [r + r / ((1+r) ^ term - 1)] x amount

El interés a aplicar dependerá del target del usuario.

Ejemplo:

Para pagar un préstamo de $ 1000 al 5% de interés durante 12 meses de un usuario PREMIUM, la ecuación sería: Cuota (mensual) = [(0.05 / 12) + (0.05 / 12) / ((1+ (0.05 / 12)) ^ 12 - 1)] x 1000 Cuotas (mensual) = $ 85.60

2. *Lista de préstamos.*

Obtener la lista de préstamos, y permitir filtrar en función de las fechas.
Además agregar paginado.

Query Param:

- from: fecha desde.
- to: fecha hasta.

Response:


```
[
    {
        "id": 1,
        "amount": 1000,
        "term": 12,
        "rate": 0.05,
        "user_id": 12,
        "target": "PREMIUM",
        "date": "2021-08-05 02:18Z",
    },
    {
        "id": 2,
        "amount": 1000,
        "term": 12,
        "rate": 0.10,
        "user_id": 3,
        "target": "MEDIUM",
        "date": "2021-08-05 02:18Z",
    }
]
```


- date: cuando se solicitó el préstamo (fecha de origen como una cadena ISO 8601).

3. *Registro de pagos realizados*

Crea un registro de un pago realizado

Payload:

- amount: monto del pago realizado (puede ser inferior o superior al monto de la cuota mensual)

Payload:

```
{
    "amount": 85.60
}
```

Response:

```
{
    "id": 2,
    "loan_id": 1,
    "debt": 400
}
```

donde
id: es el id del pago,
loan_id: es el identificador del préstamo y
debt: es el monto de la deuda

4. Obtener la deuda

Se necesita conocer la deuda pendiente de los préstamos(es decir, la deuda aún por pagar), tanto de un préstamo particular como del total o por target.

Deuda de un préstamo:

- Puede consultar la deuda de un préstamo actual o hasta una fecha dada.

Query Param:

- date: hasta esta fecha.

Response:

balance: deuda pendiente del préstamo hasta la fecha especificada en el query string.

Ejemplo de la respuesta:

```
{
    "balance": 40
}
```

Deuda total:

- Puede consultar la deuda total de los préstamos hasta una fecha determinada (igual que en el caso anterior) o por target de usuario.

Query Param:

- date: hasta esta fecha.
- target: Target (PREMIUM, MEDIUM, NEW)

Response:
Igual al caso anterior.

Consideraciones para la implementación:

- Se sugiere crear un seed de base de datos que contenga una lista precargada de usuarios con el target correspondiente.
- A la hora de modelar las condiciones de negocios aplicables según el TARGET tener en cuenta que esto debe ser fácilmente modificable/mantenible.
- No es necesario crear un endpoint para dar de alta usuarios.
- Un pago nunca puede ser mayor a la deuda de un préstamo, en caso que ocurra el sistema debería retornar error.

Tecnologías y aclaraciones:

- Java y Spring Boot (Preferentemente), pero se puede utilizar cualquier stack tecnológico que quieras en su lugar.
- Deberás subir el código a un repositorio git, preferentemente GitHub.
- Proveer el Postman collection para evaluar los endpoints (Opcional)
- Deployar la solución en algún cloud: aws, gcp o cualquier otro. O proveer un docker-compose correctamente documentado para testear la solución.
- Base de datos en H2 o con tecnología de containers (especificar en el readme la elección)

Recomendaciones

- Frameworks que usas / Arquitectura de la API
- Patrones de diseño
- Nomenclaturas de variables, métodos, clases e interfaces
- Documentación en el código (comentarios / javadocs)
- Responsabilidad de Clases
- Manejo de errores
- Testing / Cobertura
- Readme en github para contar lo que hiciste y cómo lo podemos ejecutar y probar.
- Uso de Github (commits, branches, PRs, etc). Por ejemplo, podés usar gitflow.
- Cualquier otra buena práctica que quieras considerar.
- Uso de docker preferentemente.