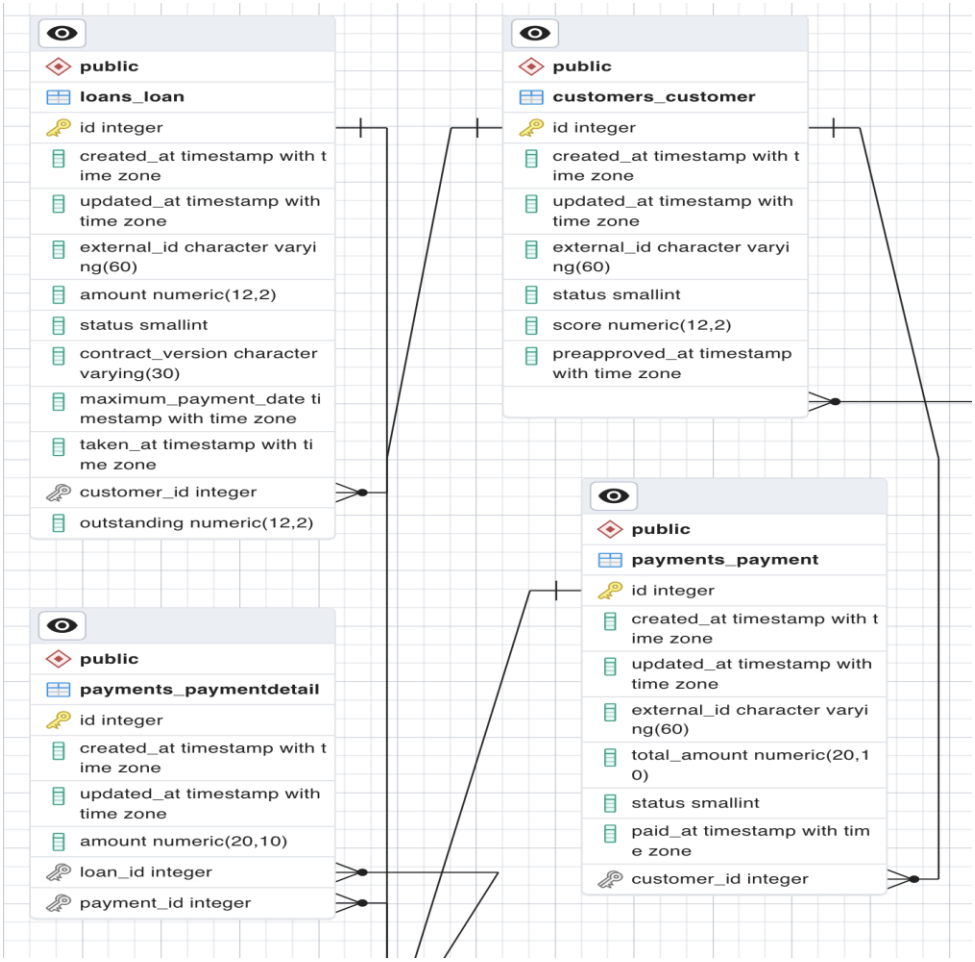
Backend Technical Test

Se solicita la creación de una API utilizando Django Rest Framework para administrar y supervisar los préstamos y pagos de los clientes. La aplicación debe incluir los siguientes modelos para llevar a cabo este control:



Customers

En este modelo, se realizará el seguimiento de todos los clientes que sean cargados. Estos clientes pueden ser cargados a través de un archivo plano o creados mediante un endpoint.

Cada cliente estará identificado por un external_id único. Además, se asignará a cada cliente un score que representa la línea de crédito otorgada, es decir, el monto que pueden utilizar para solicitar préstamos. Por último, se registrará el estado del cliente, que puede ser "Activo" (1) o "Inactivo" (2).

Loans

Este modelo se utiliza para almacenar los préstamos creados por los clientes, asegurándose de que el total del monto (outstanding) de los préstamos no exceda el límite de crédito establecido en el modelo anterior.

external_id: identificador único del préstamo.

amount: monto con el que se creó el préstamo.

contract_version: campo opcional que puede contener cualquier valor.

status:

1: "pending" (pendiente) - el estado predeterminado cuando se crea el préstamo.

2: "active" (activo) - una vez que se activa mediante un servicio, se debe actualizar la fecha de "taken_at".

3: "rejected" (rechazado) - el préstamo puede ser rechazado solo si está en el estado 1 ("pending").

4: "paid" (pagado) - una vez que el préstamo ha sido completamente pagado y el valor de "outstanding" sea 0, se debe cambiar el estado a "paid".

Además, se puede utilizar el campo "outstanding" para llevar un registro del monto pendiente de pago del préstamo.

Payments

Este modelo se utiliza para recibir pagos de los clientes, pero solo se deben aceptar pagos si el cliente tiene deudas, es decir, si tiene préstamos activos. Los pagos no deben exceder el monto de la deuda.

- external_id: identificador único del pago.
- total_amount: monto del pago.

- status: estado del pago.
    - 1: "completed" (completado).
    - 2: "rejected" (rechazado).

  Al guardar el pago, se debe tener en cuenta lo siguiente:

- Se debe almacenar el detalle del pago, ya que un pago puede cubrir uno o varios préstamos (payments_loan_detail).
- A medida que se reciben pagos, se debe disminuir el valor de "outstanding" del préstamo. Si el monto pendiente (outstanding) llega a 0, el estado del préstamo debe cambiar a "paid".
- Si se rechaza un pago, se debe actualizar el monto pendiente (outstanding) del préstamo y cambiar el estado a "active" (activo). (Extra)

  Estos son los aspectos relevantes al momento de trabajar con este modelo.

A continuacion los requerimientos por modulo;

Customers

1. Servico para crear, consultar. El cliente se debe crear con status activo.
2. Servicio para obtener el balance del cliente. Este servicio debe devolver el total de la deuda del cliente y el monto disponible. Total debt, es la suma de los outstanding de todos los prestamos pendientes y activos. Available amount, es el score menos el total debt.
3. Pruebas unitarias. (Opcional)

Ejem. Response de los servicios.

Create and get


```
{
"external_id": "external_01",
"status": 2,
"score": 4000.0,
"preapproved_at": "2023-02-12T22:29:27.177914Z"
}
```

Get customer balance

{

"external_id": "external_01",

"score": 4000.0,

"available_amount": 500.0,

"total_debt": 3500.0

}

Loans

1. Servicio para crear, consultar prestamo. El prestamo se debe crear con status activo.
2. Pruebas unitarias. (Opcional)

Ejem servicios.

Create loan

{

"external_id": "external_01-01",

"customer_external_id": "external_01",

"amount": 500.0,

"outstanding": 500.0,

"status": 0

}

Get loans by customer

[

{

"external_id": "external_01-01",

"customer_external_id": "external_01",

"amount": 500.0,

"outstanding": 500.0,

"status": 0

},

{

"external_id": "external_01-02",

"customer_external_id": "external_01",

"amount": 500.0,

"outstanding": 500.0,

"status": 0

},

]

Payments

1. Servicio para crear, consultar pagos por cliente. Se debe incluir los prestamos en los que se disperso el pago.
2. Al momento de crear el pago, rechazar pagos si son superiores a la deuda.
3. Pruebas unitarias. (Opcional)

Get payments by customer

{

"external_id": "external_01-02",

"customer_external_id": "external_01",

“loan_external_id”: “loan-01”,

'payment_date', “2023-06-12”

'status': 1

'total_amount': 1000

'payment_amount': 500

}

Puntos a tener en cuenta para la calificación:

1. Legibilidad y documentación del código: Se evaluará la claridad y organización del código, así como la presencia de comentarios y documentación adecuada para facilitar la comprensión del mismo.
2. Implementar api key autenticacion.
3. Dockerización de la aplicación: Se valorará si se ha implementado la contenerización de la aplicación utilizando Docker, lo cual permite una fácil configuración y despliegue en diferentes entornos. (Opcional)
4. Pruebas unitarias: Se espera que se hayan desarrollado pruebas unitarias para verificar el funcionamiento correcto de los diferentes componentes de la aplicación. Esto demuestra un enfoque de desarrollo orientado a la calidad y la detección temprana de errores. (Opcional)
5. Documentación de la API: Será evaluada la presencia de una documentación clara y completa de la API. Esto puede incluir el uso de herramientas como Postman o Swagger para describir los endpoints, parámetros, respuestas y autenticación. (Opcional)
6. Aplicación de principios SOLID: Se evaluará si se han aplicado los principios SOLID en el diseño de la aplicación. (https://www.oscarblancarteblog.com/2018/08/15/principios-solid-patrones-diseno/) (Opcional)

Lo ideal es completar la prueba, sin embargo, no es obligatorio. Una vez finalizado, se debe subir el código fuente a un repositorio público y compartirlo. Esto permitirá que otros puedan acceder al código y revisar el trabajo realizado.

Dependiendo de la finalización de la prueba, así como de la implementación de los puntos extra mencionados, se determinará el nivel técnico de la persona evaluada. La finalización exitosa de la prueba y la implementación de los puntos extra demuestran habilidades y conocimientos adicionales, lo que puede reflejar un mayor nivel de competencia técnica.