

NTTDATA Bootcamp Microservicios

Proyecto I

El sistema a desarrollar está planteado en el contexto del negocio bancario que a medida que se va avanzando en los proyectos, se irá ampliando en base a este mismo proyecto.

# Bases a Desarrollar

|  |  |  |
| --- | --- | --- |
| Actividad | Porcentaje | Estado de Actividad |
| * Desarrollo de microservicios con Java 8. | 100% | TERMINADO |
| * Utilizar Spring Boot Webflux como framework base. | 100% | TERMINADO |
| * El proyecto debe utilizar Maven como manejadores de dependencias. | 100% | TERMINADO |
| * Los microservicios proporcionados deben implementar controladores REST. | 100% | TERMINADO |
| * Usar el patrón database per service, por lo que un microservicio no puede tocar ninguna tabla o colección que utilice otro microservicio. | 10% | PENDIENTE |
| * Utilizar inyección de dependencias. | 100% | TERMINADO |
| Utilizar propiedades de configuración externalizadas con un Config Server. | 10% | PENDIENTE |
| * Los nombres de las clases, métodos y las URLs deberán estar en inglés. | 100% | TERMINADO |
| * La base de datos a utilizar será MongoDB. | 100% | TERMINADO |
| * Uso de Lombok para reducir código. | 100% | TERMINADO |
| * Manejo de trazas con Logback y utilizar el nivel del log adecuado. | 100% | TERMINADO |

# Funcionalidades del sistema

|  |  |  |
| --- | --- | --- |
| Actividad | Porcentaje | Estado de Actividad |
| * El sistema debe manejar la información de los clientes de un banco. | 100% | TERMINADO |
| * Los clientes del banco son de dos tipos: personal o empresarial. | 100% | TERMINADO |
| * El sistema debe manejar la información de los siguientes productos que ofrece el banco:   + Pasivos (cuentas bancarias)     - Ahorro: libre de comisión por mantenimiento y con un límite máximo de movimientos mensuales. | 20% | PENDIENTE |
| * El sistema debe manejar la información de los siguientes productos que ofrece el banco:   + Pasivos (cuentas bancarias)     - Cuenta corriente: posee comisión de mantenimiento y sin límite de movimientos mensuales. | 20% | PENDIENTE |
| * El sistema debe manejar la información de los siguientes productos que ofrece el banco:   + Pasivos (cuentas bancarias)     - Plazo fijo: libre de comisión por mantenimiento, solo permite un movimiento de retiro o depósito en un día específico del mes. | 20% | PENDIENTE |
| * El sistema debe manejar la información de los siguientes productos que ofrece el banco:   + Activos (créditos)     - Personal: solo se permite un solo crédito por persona. | 20% | PENDIENTE |
| * El sistema debe manejar la información de los siguientes productos que ofrece el banco:   + Activos (créditos)     - Empresarial: se permite más de un crédito por empresa. | 20% | PENDIENTE |
| * El sistema debe manejar la información de los siguientes productos que ofrece el banco:   + Activos (créditos)     - Tarjeta de Crédito personal o empresarial. | 20% | PENDIENTE |
| * Un cliente personal solo puede tener un máximo de una cuenta de ahorro, una cuenta corriente o cuentas a plazo fijo. | 20% | PENDIENTE |
| * Un cliente empresarial no puede tener una cuenta de ahorro o de plazo fijo pero sí múltiples cuentas corrientes. | 20% | PENDIENTE |
| * Las cuentas bancarias empresariales pueden tener uno o más titulares y cero o más firmantes autorizados. | 20% | PENDIENTE |
| * Un cliente puede tener un producto de crédito sin la obligación de tener una cuenta bancaria en la institución. | 20% | PENDIENTE |
| * Un cliente puede hacer depósitos y retiros de sus cuentas bancarias. | 20% | PENDIENTE |
| * Un cliente puede hacer pagos de sus productos de crédito. | 20% | PENDIENTE |
| * Un cliente puede cargar consumos a sus tarjetas de crédito en base a su límite de crédito. | 20% | PENDIENTE |
| * El sistema debe permitir consultar los saldos disponibles en sus productos como: cuentas bancarias y tarjetas de crédito. | 20% | PENDIENTE |
| * El sistema debe permitir consultar todos los movimientos de un producto bancario que tiene un cliente. | 20% | PENDIENTE |

# Requerimientos no funcionales obligatorios del sistema

|  |  |  |
| --- | --- | --- |
| Actividad | Porcentaje | Estado de Actividad |
| * Elaborar y mantener un diagrama en draw.io con el diseño de la solución. | 100% | TERMINADO |
| * El repositorio de datos deberá estar en documentos NoSQL. | 100% | TERMINADO |
| * Para el manejo de datos se deberá utilizar Spring Data y no se deberá manejar la creación de SQL dinámicos y evitar el uso de la anotación @Query. | 100% | TERMINADO |
| * Para todas las entidades de negocio se debe implementar sus operaciones CRUD: Create, FindAll, Update, Delete. | 100% | TERMINADO |
| * Crear los endpoints REST para cada una de las operaciones de los repositorios. | 100% | TERMINADO |
| * Utilizar los lineamientos REST para las operaciones CRUD. | 100% | TERMINADO |
| * El sistema no tendrá implementado ninguna interfaz gráfica, la verificación de las funcionalidades se realizarán utilizando Postman. | 100% | TERMINADO |

# Recomendaciones y Consideraciones

* Realicen primero las funcionalidades obligatorias.
* Realicen primero las funcionalidades opcionales más sencillas.
* No deben tener configuraciones en el código.
* Las clases y los métodos deben estar comentados.
* El uso de lambdas y streams de Java 8 es deseable.
* Deben subir su código a un repositorio git en github.
* Cada grupo deberá presentar su propia solución.

# Artefactos y entregables.

* Todos los días laborables deberán subir el avance de su desarrollo a sus repositorios en github.
* Cada microservicio deberá tener su propio repositorio.
* La entrega del código de este proyecto tiene como fecha fin de entrega el 8 de agosto hasta las 18 horas con el código que esté en ese momento en el repositorio remoto.