From: Manuel Figueira

To: You
Subject: Bienvenida al proyecto Home Banking

¡Bienvenido equipo al proyecto de Home Banking!

Nuestro banco, Mindhub Brothers, tiene una historia de más de 50 años en el mercado, brindando estabilidad y respaldo a millones de clientes sobre sus bienes así como también múltiples servicios y préstamos a tasas muy competitivas.

En la actualidad los clientes buscan realizar sus operaciones bancarias lo más rápido posible y sin tener que pasar por muchos inconvenientes, es por ello que en pro de mantener la calidad y el buen servicio que provee el Banco Vinotinto deseamos crear un sistema que en una primera etapa provea algunas de las funcionalidades que actualmente se pueden realizar de manera presencial en el banco, si todo va bien podremos seguir agregando más.

Por lo tanto nos gustaría que construyan una aplicación que permita realizar operaciones bancarias desde cualquier dispositivo, pc, teléfono, tablet a través de una aplicación web. Si la aplicación web tiene éxito podríamos pensar en crear una aplicación móvil que utiliza los mismo servicios.

Estamos muy emocionados de ver qué nos pueden presentar.

¡Gracias!
Manuel
Mindhub Brothers Bank

From: Laureano Andreotti
To: You
Subject: Plan del proyecto Home Banking

Equipo,

Asumo que recibieron el correo de Manuel. Estamos muy contentos de recibir este proyecto de un nuevo cliente así que debemos dejar una buena impresión para que nos asignen más proyectos en el futuro.

Luego de analizar el requerimiento y tener en cuenta las necesidades futuras he llegado a la conclusión de que necesitamos seguir una arquitectura de cliente/servidor, por un lado tendremos el código de front-end - cliente (como aplicaciones web, aplicaciones nativas móviles) que tendrá toda la parte visual y las interacciones con el usuario y por otro el código de back-end - servidor que administrará las cuentas de los clientes del banco, por ejemplo el estado de cuenta, transacciones realizadas, préstamos adquiridos y demás operaciones, la comunicación entre ámbas partes se llevará a cabo a través de peticiones HTTP enviando y recibiendo la información en formato JSON. De esta manera podemos en el futuro crear otras aplicaciones que utilicen el mismo back-end - servidor sin tener que cambiar nada y eligiendo la tecnología de front-end que se desee.

Como nos piden en primera instancia una aplicación web para interactuar con el servidor, usaremos HTML 5, CSS y javascript, utilizando Vue.js, Bootstrap y Axios para facilitar el desarrollo. Esto permitirá a los clientes acceder desde cualquier dispositivo desde un navegador web, así los clientes con dispositivos móviles podrán operar hasta que se decida crear una aplicación nativa.

Para el backend utilizaremos Java con el framework Spring Boot que nos permite crear aplicaciones de manera rápida y sencilla, utilizaremos el módulo Spring JPA para eliminar la necesidad de implementar consultas SQL directamente en el código, solo necesitaremos los repositorios de cada entidad. Además el módulo Spring MVC nos provee una manera de crear facilmente servicios REST que permiten enviar y recibir información a través de peticiones HTTP, cualquier programa que pueda enviar peticiones HTTP (como los navegadores) podrá acceder al back-end - servidor.

Como primer paso debemos configurar el ambiente de desarrollo y crear el esqueleto de la aplicación, utilizaremos la herramienta spring initializr para realizar ese proceso de manera sencilla, además se debe crear la primera entidad llamada Client y crear un cliente de prueba en la base de datos. Una vez que se termine de realizar el esqueleto de la aplicación se debe enviar un reporte de las herramientas instaladas.

¡Gracias!
Laureano

From: Laureano Andreotti
To: You
Subject: Creando la vista de cuentas

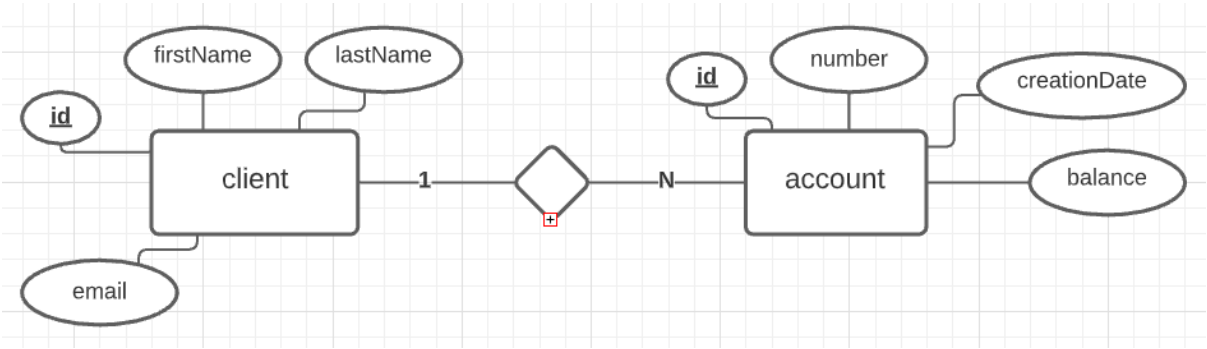
Muy bien, hemos avanzado en la construcción del proyecto de Home Banking y ya tenemos el esqueleto de la aplicación, ahora necesitamos agregar todas las funcionalidades que restan comenzando por crear las entidades del modelo de datos así como las interfaces para mostrarlos como ver las cuentas, las transacciones, los préstamos, etc.. para luego implementar la habilidad de poder crear cuentas, realizar transacciones, pedir préstamos, etc…

Siguiendo la Metodología Ágil, los objetivos se plantean en historias de usuario y pruebas de aceptación.

A continuación la primera historia de usuario:

> **User Story 1**: como un cliente, ver las cuentas asociadas para ver la cantidad de dinero que tienen.
>
> **Acceptance test:**
> **Con** el cliente Melba Lorenzo y las cuentas VIN001 y VIN002
> **Entonces** ver en una página accounts.html el nombre del cliente y las cuentas con el saldo disponible.

En el siguiente diagrama puedes ver cómo es la relación de **uno a muchos** entre cliente y cuenta, recuerda que con Spring y Java Persistence API (JPA) no necesitas crear la tabla cuenta directamente en la base de datos, solo debes crear la clase e indicar que es una entidad así como crear el repositorio correspondiente.



Ya sabemos que Spring REST Repositories crea una API web RestFull que permite ver y crear entidades haciendo peticiones HTTP, el problema es que esta API no es muy apropiada para una aplicación ya que se deben hacer muchas peticiones para mostrar los datos a los usuarios, así que implementaremos una API REST más amigable para nuestros propósitos. El esquema a trabajar es:

- La API REST Hateos creada automáticamente por los repositorios será accedida desde la url **/rest**, por ejemplo si antes era **/clients** o **/clients/1** ahora será **/rest/clients** o **/rest/clients/1**
- La API REST personalizada creada por nosotros será accedida desde la url **/api**, por ejemplo **/api/clients /api/clients/1**
- Las páginas web serán accedidas desde la url **/web**, por ejemplo **/web/accounts.html**

Cuando tengas todos estos cambios envíame la carpeta del proyecto, recuerda ejecutar la tarea clean antes de crear el archivo .zip.

¡Suerte y saludos!.
Laureano

From: Laureano Andreotti
To: You
Subject: Creando la vista de transacciones

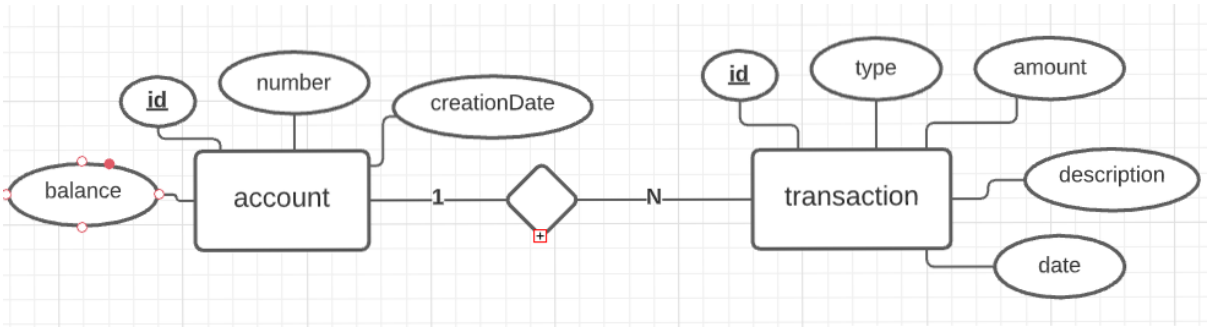
Saludos!,

Ya tenemos los clientes y las cuentas, es momento de agregar las transacciones.

En los bancos los clientes pueden tener varias cuentas y por cada cuenta se pueden realizar muchas transacciones, en la fase 2 del proyecto agregaremos la capacidad de que el cliente pueda crear transacciones, por ahora necesitamos:

- Crear la entidad Transaction y su repositorio para guardar transacciones en las cuentas.
- Crear una página web para mostrar las transacciones de una cuenta

En el siguiente diagrama puedes ver cómo es la relación de **uno a muchos** entre cuenta y transacciones:



La historia de usuario es:

> **User Story 2**: como un cliente, entrar en una cuenta para poder ver las transacciones.
>
> **Acceptance test:**
> **Con** el cliente Melba y la cuenta VIN001
> **Entonces** ver en una página **account**.html?**id**=1 el listado de transacciones de la cuenta VIN0001 así como la información de la cuenta.
> **Con** el cliente Melba y la cuenta VIN002
> **Entonces** ver en una página **account**.html?**id**=2 el listado de transacciones de la cuenta VIN0002 así como la información de la cuenta.

Ten en cuenta que la página web es la misma "**account.html**" lo que cambia es el valor del parámetro de la petición llamado **id**.

Las transacciones de crédito deben mostrarse de color verde y las transacciones de débito deben mostrarse de color rojo.

Cuando termines envíame el zip con el proyecto.

¡Que te diviertas!
Laureano.

From: Laureano Andreotti
To: You
Subject: Creando la vista de préstamos

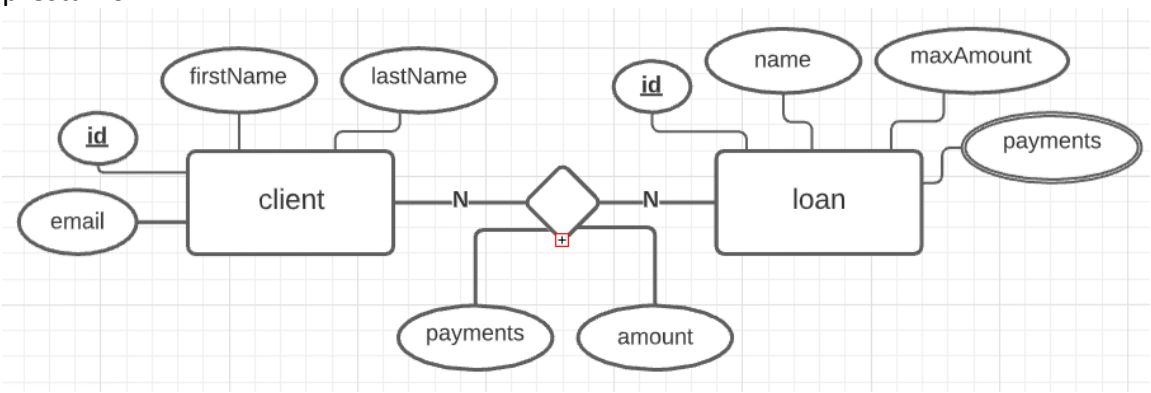
Saludos!,

Genial, buen trabajo con las cuentas. Ahora se deben implementar los préstamos, al igual que en el proceso anterior solo nos enfocaremos en la construcción del modelo de datos y las vistas para mostrarlo, luego en la segunda fase de desarrollo agregaremos funcionalidad a la aplicación para que los clientes puedan solicitar un préstamo.

A continuación la historia de usuario:

> **User Story 3**: como un cliente, poder ver los préstamos solicitados para ver el monto y las cuotas solicitadas de cada uno.
>
> **Acceptance test:**
> **Con** el cliente Melba
> **Entonces** ver en la página account.html el nombre del cliente y los préstamos con el monto solicitado.

En el siguiente diagrama puedes ver la relación de **muchos a muchos** entre cliente y préstamo:



La relación debe ser muchos a muchos ya que un cliente puede solicitar muchos préstamos y un préstamo puede ser solicitado por muchos clientes. A diferencia de las transacciones en que existe una transacción única asociada por cuenta, en este esquema el préstamo define las bases de uno, como el monto máximo que se permite solicitar, la cantidad de pagos en que se puede pagar y el nombre. Es cuando se asocia a un cliente que se define el monto exacto solicitado por el cliente y en cuantas cuotas lo pagará. Así no tiene que repetirse toda la información del préstamo por cada cliente.

Cuando termines envíame el zip con el proyecto.
¡Éxito!

Laureano.

From: Laureano Andreotti
To: You
Subject: Creando la vista de tarjetas

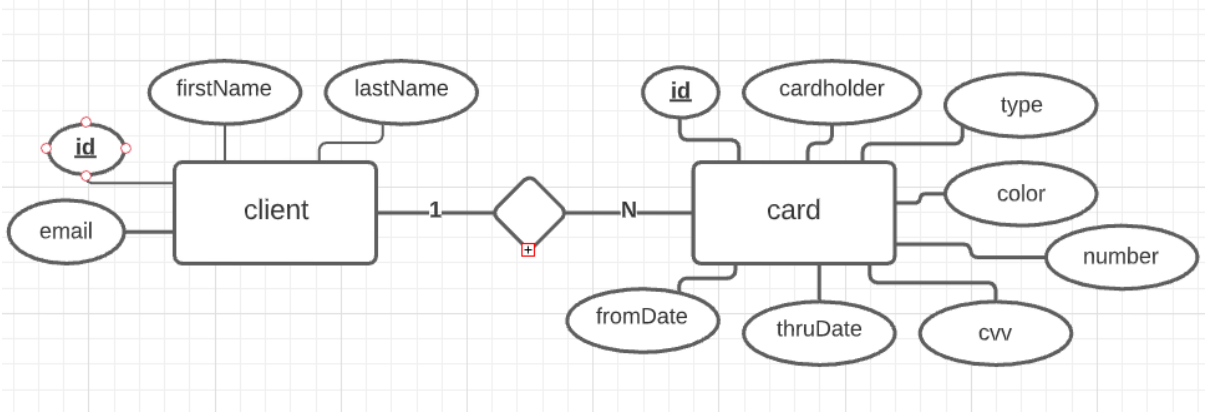
Saludos!,

Ya casi tenemos todo el modelo de datos creado, solo nos falta implementar las tarjetas. Como ya tienes experiencia creando las entidades de seguro no tendrás muchos problemas en llevar a cabo este requerimiento. Lo que se necesita es que cada cliente pueda tener una o más tarjetas, cada tarjeta tendrá un tipo (CREDIT, DEBIT), número, un código de seguridad, fecha de validez (desde, hasta), el nombre del tarjetahabiente y color (GOLD,SILVER,TITANIUM).

A continuación la historia de usuario:

> **User Story 4**: como un cliente, poder ver las tarjetas asociadas para ver su número, validez, proveedor, tipo y su código de seguridad.
>
> **Acceptance test:**
> **Con** el cliente Melba
> **Entonces** ver en la página cards.html el nombre del cliente y las tarjetas con su información.

En el siguiente diagrama puedes ver la relación de **uno a muchos** entre cliente y tarjetas:



Como las tarjetas se deben mostrar en un html aparte (cards.html) es necesario crear un menú, el mismo tendrá un enlace hacia accounts.html y otro hacia cards.html. De nuevo, el diseño del menú quedará a tu elección, usualmente se posiciona arriba de la página o a uno de los lados. Para impresionar al cliente sería genial si podemos ¡mostrar las tarjetas como lo son en verdad!

Cuando termines envíame el zip con el proyecto.
¡Que te diviertas!
Laureano.

From: Laureano Andreotti
To: You
Subject: HomeBanking: comienzo fase 2

Saludos!,

La estructura principal del proyecto ya está terminada, tenemos el modelo de datos construido, servicios REST para obtener información de la base de datos y algunas vistas creadas para mostrar el contenido de manera amigable para el usuario. Es tiempo de agregar funcionalidades de banca, como crear cuentas, realizar transacciones y solicitar préstamos.

En el Modelo MVC (modelo, vista, controlador) has implementado la M y la V, Modelo porque ya tienes la capa de datos, que son las entidades y los repositorios, así como también servicios que permiten obtenerlos y Vista porque has implementado distintas páginas web para presentar la información al usuario. Ahora debes agregar controladores, es decir botones y demás cosas que permitan al cliente realizar operaciones, en el trasfondo con javascript tendremos llamadas a servicios del backend que realizan dichas acciones.

Antes que nada tenemos que permitir a los clientes poder registrarse, te enviaré los detalles en otro email.

¡Muy bien!
Laureano.

From: Laureano Andreotti
To: You
Subject: Implement client registration and login

Saludos!,

Es hora de que se puedan crear cuentas, hacer transacciones y solicitar préstamos, pero para hacer eso los usuarios deben estar registrados y tener una sesión iniciada. Registrarse significa crear un usuario del sistema e iniciar sesión es que un usuario creado cree una sesión temporal de trabajo.

Para indicar lo más sencillo el requerimiento se presentan las historias de usuario a implementar:

User Story 5: como un cliente, poder iniciar sesión para acceder al homebanking

Acceptance test:
Con el cliente Melba Lorenzo,
Entonces llenar los datos del formulario de inicio de sesión (mostrar error si no se coloca alguno), pulsar el botón **sign in** e iniciar sesión, proceder a ver la pantalla de cuentas.

User Story 6: como un nuevo cliente, poder registrarse para iniciar sesión y acceder al homebanking

Acceptance test:
Con el cliente Rodrigo Ribeiro,
Entonces llenar los datos del formulario de registro (mostrar error si no se coloca alguno), pulsar el botón **sign up** e iniciar sesión, proceder a ver la pantalla de cuentas.

User Story 7: como un cliente con sesión iniciada, poder cerrar sesión para bloquear el acceso al homebanking.

Acceptance test:
Con el cliente Rodrigo Ribeiro o Melba Lorenzo con sesión iniciada
Entonces pulsar el botón **sign out**, cerrar sesión y volver al inicio

Para mantener la consistencia creé una tabla que muestra las url disponibles de la API.

URL	Efecto en el servidor	Respuesta al cliente
GET /api/clients	ninguno	JSON con listado de clientes
GET /api/clients/{id}	ninguno	JSON con los datos de un cliente
GET /api/clients/current	ninguno	JSON con los datos del cliente autenticado
POST /api/login parámetros: email, password	sesión creada	éxito: 200 respuestas de error: 401 unauthorized si no existe el usuario o la contraseña no coincide
POST /api/logout	Si existe una sesión creada entonces se cierra	éxito: 200
POST /api/clients parámetros: firstName, lastName, email, password	Nuevo cliente creado con los parámetros recibidos Sesión iniciada	éxito: 201 created respuestas de error: 403 forbidden, si el email ya existe

Cuando termines envíame el zip con el proyecto.

¡Que te diviertas!
Laureano.

From: Laureano Andreotti
To: You
Subject: Permitir crear cuentas

Saludos!,

Muy buen trabajo, ya los clientes se pueden registrar y hemos agregado la capa de seguridad a la aplicación. El siguiente paso es permitir a los clientes crear cuentas, además al registrarse sería bueno que se genere una cuenta de manera automática, ya que un cliente registrado sin cuenta no tiene mucho sentido.

Una cosa a tener en cuenta es que los clientes solo podrán tener como máximo 3 cuentas.

Además de poder crear cuentas también se deben poder crear tarjetas, para crear una tarjetas por lo que también debes implementar esa funcionalidad.

A continuación las historias de usuario:

User Story 8: como un cliente con sesión iniciada, poder crear cuentas para realizar operaciones.

Acceptance test:
Con el cliente Melba Lorenzo con sesión iniciada
Entonces ir a la página accounts.html y pulsar el botón **create**, seguidamente ver la cuenta creada en pantalla.

User Story 9: como un cliente con sesión iniciada, poder crear tarjetas de débito o crédito para luego utilizarlas.

Acceptance test:
Con el cliente Melba Lorenzo con sesión iniciada
Entonces ir a la página create-cards.html, indicar el tipo de tarjeta y el color, luego pulsar el botón **create**, seguidamente ver la tarjeta creada en pantalla.

Se presenta la tabla con las URL de la API:

URL	Efecto en el servidor	Respuesta al cliente
POST /api/clients/current/accounts	cuenta creada y asignada al cliente autenticado	éxito: 201 created respuestas de error: 403 forbidden, si el cliente ya tiene 3 cuentas creadas
GET /api/clients/current/accounts	ninguno	JSON con las cuentas de un cliente
POST /api/clients/current/cards	tarjeta creada y asignada al cliente autenticado	éxito: 201 created respuestas de error: 403 forbidden, si el cliente ya tiene 3 tarjetas creadas por tipo
GET /api/clients/current/cards	ninguno	JSON con las tarjetas de un cliente

Cuando termines envíame el zip con el proyecto.

¡Que te diviertas!
Laureano.

From: Laureano Andreotti
To: You
Subject: Permitir realizar transferencias

Muy buen trabajo con la creación de cuentas y tarjetas, estamos progresando mucho con el sistema. El siguiente requerimiento a implementar es la posibilidad de que un cliente pueda transferir dinero de su cuenta a otra.

La transferencia de dinero es un proceso sencillo, se resta dinero de una cuenta y se suma dinero a otra, es decir se deben crear dos transacciones una para la cuenta de origen restando dinero y otra para la cuenta de destino sumando el dinero. Lo importante acá es asegurar que el proceso se cumpla, es decir que si se resta dinero en una cuenta obligatoriamente se debe sumar en la otra, en caso de que ocurra un error en alguno de esos pasos se debe revertir la operación, esto se conoce como transacción.

Como la transferencia de fondos es un tema delicado, el nuevo servicio debe verificar muchas cosas antes de realizarla, tanto de forma como de seguridad.

A continuación la historia de usuario:

> **User Story 10**: como un cliente con sesión iniciada, poder realizar una transacción para transferir dinero de una cuenta a otra.
>
> **Acceptance test:**
> **Con** el cliente Melba Lorenzo con sesión iniciada
> **Entonces** ir a la página **transfers.html**, seleccionar la cuenta de origen, indicar la cuenta de destino, monto y pulsar el botón **transfer**, aceptar la notificación de seguridad preguntando si se quiere transferir el dinero, una vez completada la transferencia se debe mostrar un mensaje indicando que fue exitosa.

Se presenta la tabla con las URL de la API:

URL	Efecto en el servidor	Respuesta al cliente
POST /api/transactions	transacción creada para cuenta de origen y transacción creada para cuenta de destino	éxito: 201 created respuestas de error: 403 forbidden, si el monto o la descripción estan vacios 403 forbidden, si alguno de los numeros de cuenta estan vacios 403 forbidden, si la cuenta de origen no existe

		403 forbidden, si la cuenta de destino no existe 403 forbidden, si la cuenta de origen no pertenece al cliente autenticado 403 forbidden, si el cliente no tiene fondos 403 forbidden, si la cuenta de origen es la misma que la destino

Cuando termines envíame el zip con el proyecto.

¡Mucha suerte!
Saludos!,

Laureano

From: Laureano Andreotti
To: You
Subject: Permitir solicitar préstamos

Excelente trabajo, ya se pueden realizar transferencias de manera cómoda e intuitiva, ¡casi terminamos con la primera entrega del sistema!. El cliente va a estar muy contento.

Esta es la última funcionalidad core del sistema de homebanking que implementaremos, para solicitar un préstamo el cliente debe seleccionar uno de los préstamos disponibles y luego indicar monto y en cuantas cuotas lo pagará, sería genial que se le pueda presentar el resumen con el monto de las cuotas que tendrá que pagar antes de crear el préstamo. Al solicitar el préstamos automáticamente se acreditará en la cuenta de destino el monto solicitado (son créditos preaprobados ;D).

Como es una operación que involucra más de una operación en las que se deben crear la solicitud del préstamo, luego acreditar el monto en la cuenta de destino y crear una transacción que indique el movimiento, entonces este proceso debe ser una transacción.

A continuación la historia de usuario:

> **User Story 11**: como un cliente con sesión iniciada, poder solicitar un préstamo para tener dinero disponible en una cuenta.
>
> **Acceptance test:**
> **Con** el cliente Melba Lorenzo con sesión iniciada
> **Entonces** ir a la página **loan-application.html**, seleccionar uno de los préstamos disponibles, indicar el monto a solicitar, indicar las cuotas, indicar la cuenta de destino, una vez completada la solicitud se debe redirigir a la página **accounts.html**

Se presenta la tabla con las URL de la API:

URL	Efecto en el servidor	Respuesta al cliente
GET /api/loans	ninguno	Json con los préstamos disponibles
POST /api/loans	solicitud de préstamo creado al cliente autenticado transacción creada para cuenta de destino cuenta de destino actualizada con el	éxito: 201 created respuestas de error: 403 forbidden, si alguno de los datos no es válido 403 forbidden, si la cuenta de destino no existe 403 forbidden, si la cuenta de destino no pertenece al cliente autenticado

	monto	403 forbidden, si el préstamo no existe 403 forbidden, si el monto solicitado supera el monto máximo permitido del préstamo solicitado 403 forbidden, si la cantidad de cuotas no está disponible para el préstamo solicitado

Cuando termines envíame el zip con el proyecto.

¡Que te diviertas!
Laureano.

From: Laureano Andreotti
To: You
Subject: Configurar un nuevo motor de base de datos.

Muy bien, hemos terminado con la implementación del core de las operaciones bancarias que debe realizar el sistema. Ahora comenzaremos a realizar algunos cambios en la aplicación, cambios no funcionales como se llaman en el desarrollo de software. El primer cambio será hacer que la aplicación trabaje con un motor de base de datos que almacene los datos en el disco duro para que sean persistentes, el segundo

Para llevar esto a cabo primero debes aprender más sobre las bases de datos, es por eso que la empresa te habilitó un curso corto en el que aprenderás sobre modelado de datos, diagramas de entidad relación, instalación de un motor de base de datos, uso del lenguaje SQL para crear esquemas, realizar consultas, modificar datos.

Luego de finalizar el curso deberás modificar la configuración de la aplicación para que se conecte al nuevo gestor de base de datos.

¡Que te diviertas!
Laureano.