netmind
a bts company

Spring Boot

Caso Práctico

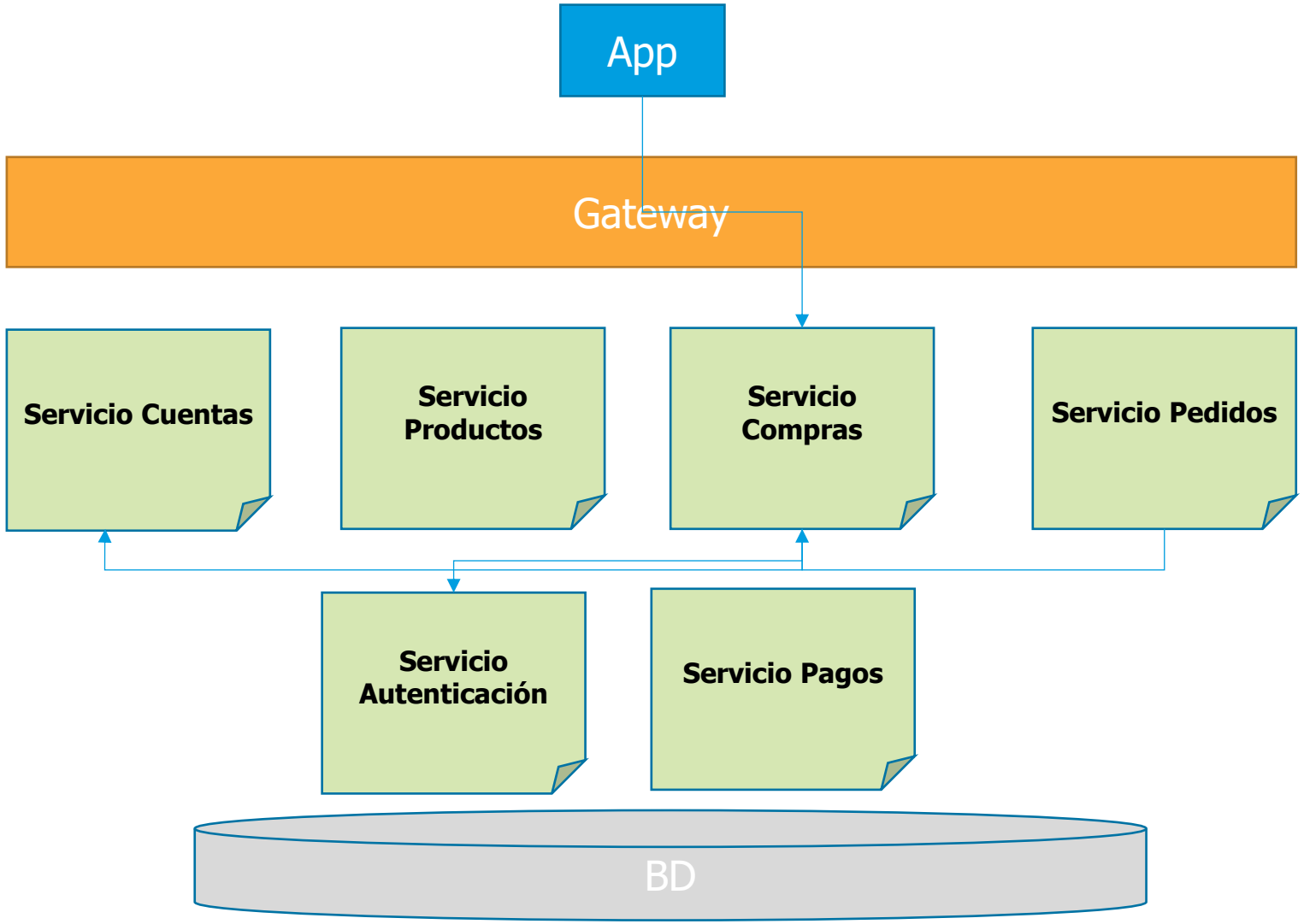
Caso práctico - MyOnlineShoppingService

- **"MyOnlineShoppingService"** es el proyecto estrella de Banana Apps.
- Es un **marketplace** que permitirá implementar una API para interactuar con el ecosistema de productos de los partners del cliente (YourProducts LLC) desde cualquier tipo de aplicación de usuario final (**multicanalidad**).
- También permitirá integrar **aplicaciones de terceros**, especialmente partners con el servicio de productos de la organización.
- Se quiere implementar un **sistema basado en microservicios**, ya que este sistema se va a implementar por distintos equipos a los que se quiere dotar de independencia y autonomía. Se prevé que el **ritmo de entrega sea diferente** para cada bloque de la aplicación. Asimismo se quiere que la aplicación sea **resilente** a los fallos de sus partes y **escale de manera independiente**.
- Tras varias reuniones con expertos del dominio de YourProducts LLC se han definido historias de usuario a nivel de negocio y requisitos para el sistema…

Caso práctico – MyOnlineShoppingService USs

- Como app cliente de la API quiero poder gestionar mi cuenta en el sistema, para actualizar mis datos cuando sea necesario.
- Como app admin de la API quiero poder gestionar las cuentas en el sistema ya sea para dar de alta, baja, modificar, apps cliente.
- Como app cliente de la API quiero poder gestionar mis productos en el sistema, para dar de alta, baja, actualizar productos cuando sea necesario.
- Como app cliente de la API quiero poder gestionar la compra de productos en el sistema, para añadir, eliminar, N productos que el usuario quiere comprar.
- Como app cliente de la API quiero poder gestionar mis pedidos para ver el histórico de los mismos.
- ….
- El acceso a todos los servicios de la API debe estar securizada con OAuth2.

01.1 Define el o los dominios y los servicios a implementar



EXERCISE

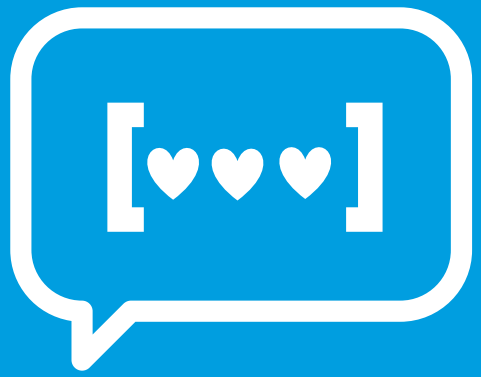
La API del servicio de Cuentas

- Se quiere que el servicio de cuentas pueda ofrecer las siguientes **funcionalidades** a través de una **API**:
  - Entregar una cuenta concreta de manera individual, validando que su id y propietario se corresponden.
  - Listar las cuentas de un usuario.
  - Crear, actualizar, borrar cuentas para un usuario concreto. Validar su correspondencia.
  - Añadir dinero al balance, indicando cuenta, cantidad y propietario.
  - Hacer un retiro, indicando cuenta, cantidad y propietario.
  - Borrar todas las cuentas de un usuario.
  - Comprobar que un usuario puede recibir un préstamo. Esto debe ser posible solo si el préstamo solicitado no supera el 80% monto de las cuentas de un usuario.
- **Diseña** un esquema para la API del servicio.
  - Define las uris y los comandos pertinentes para cada recurso o servicio.
  - Define los códigos de respuesta para los distintos escenarios.
  - Ten en cuenta que para pedir información la API puede entregarse en formato json y xml, pero la API solo aceptará json como cuerpo de las peticiones.
  - Asimismo, jerárquicamente, una cuenta debe tener siempre un propietario.
  - Se requiere que para todos los endpoints, en los recursos relacionados con las cuentas, se referencie al propietario solo por su identificador (owner_id: number).
    - **Tips**: ver opciones "*insertable*" y "*updatetable*" en *@Column* y *@JoinColumn* (*https://www.objectdb.com/api/java/jpa/Column*)

- **Implementa** los endpoints usando springboot.
  - Gestiona adecuadamente el cuerpo y código de estados.
  - Gestiona los escenarios negativos como excepciones.
  - Asegúrate de que los datos se validen siempre.
- Define la **configuración** y las **propiedades** del proyecto (yaml).
- Añade los perfiles ***dev*** y ***prod***.
  - El perfil **dev** debe usar el puerto 9900 y debe conectarse a una base de datos H2.
    - Debe precargar el conjunto de datos de desarrollo.
    - El esquema se debe generar automáticamente a partir de los mapeos de entidad
    - Debe mostrar los logs en nivel debug y se deben guardar en /var/logs/accounts.log
  - El perfil **prod** deber usar el puerto 9943 y debe conectarse a una base da datos Mysql.
    - No debe cargar datos
    - El esquema se debe generar automáticamente a partir de los mapeos de entidad
    - Debe mostrar los logs en nivel error
    - Debe tener una ruta inicial de /api/v1 para todos los endpoints

Bonus:

- En el caso de retiro de dinero, se requiere que cuando el monto de retiro supere el monto de la cuenta indicada, y en el caso de que el usuario tenga más cuentas, se descuente de estas, si es posible.
  - Si no es posible, se debe disparar una excepción.
  - Por cuestiones de eficiencia y evolutividad, este proceso se requiere que se realice en la capa de persistencia.
- Para el tipo de cuenta, se debe validar que los posibles valores textuales son "Personal" o "Company".



We would like to know your opinion!

Please, let us know what you think about the content.
From Netmind we want to say thank you, we appreciate time and effort you have taking in answering all of that is important in order to improve our training plans so that you will always be satisfied with having chosen us
quality@netmind.es

netmind
a bts company

Thanks!

Follow us: