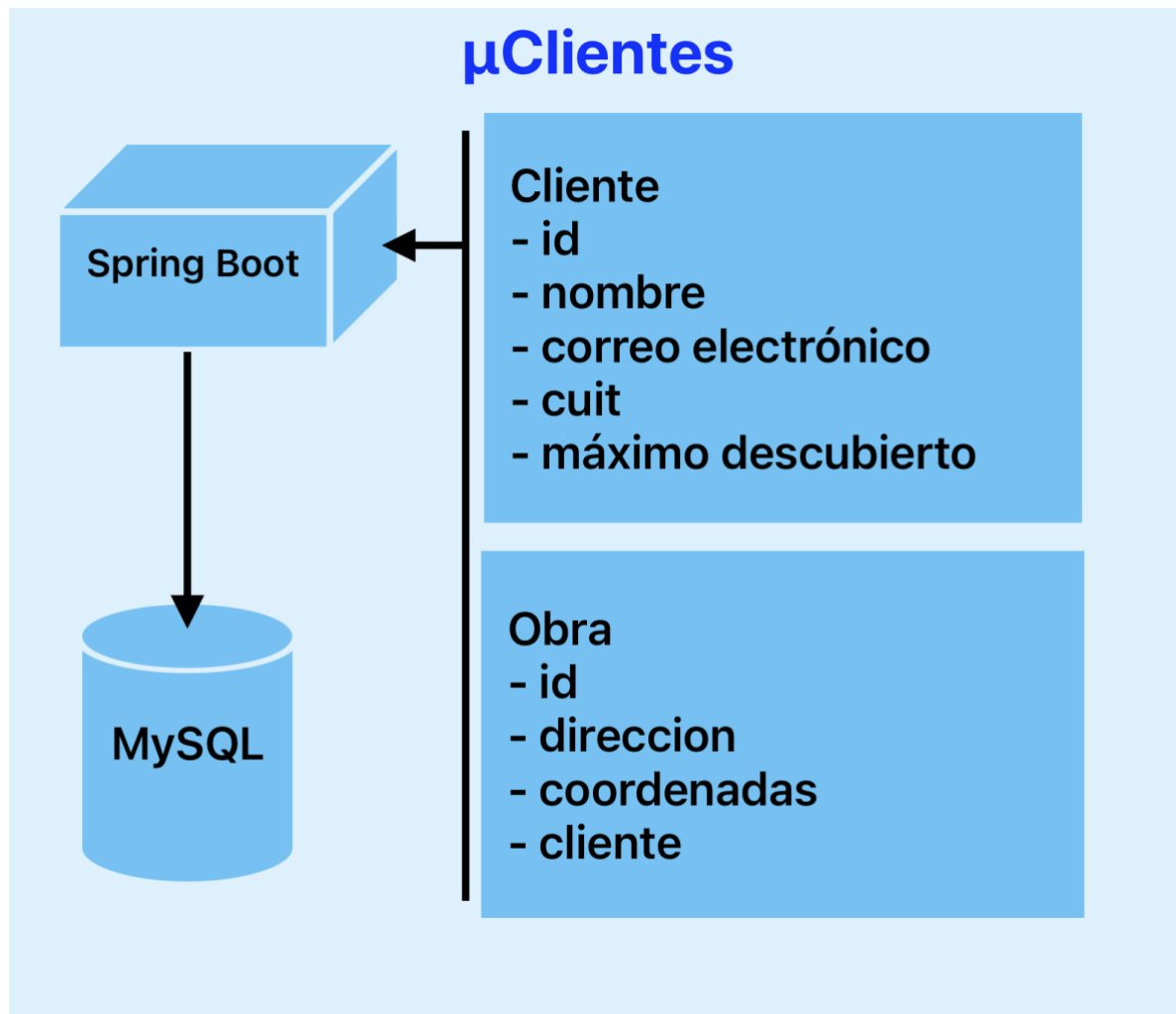


## TP FINAL DAN – MS CLientes

Modelo propuesto



### Objetivos

Gestionar el alta, baja, consulta, y actualización de clientes en el sistema y de las obras asignadas a los mismos.

### Requisitos

Se puede dar de Gestionar la información de un cliente a través de un endpoint rest con la información mínima obligatoria:

- Nombre del cliente
- Correo Electrónico
- CUIT
- Máximo Descubierto (indica la cantidad máxima de dinero que puede quedar debiendo en la cuenta corriente).
- Maximo cantidad de obras en ejecución: un valor entero que indica la cantidad máxima de obras que un cliente puede tener activas


Se puede dar de gestionar una obra con la siguiente información

- Dirección
- Coordenadas (lat lng)
- Presupuesto estimado de la obra.
- Estado:
  - Habilitada: una obra está habilitada si no sobrepasa el máximo de cantidad de obras activas. Caso contrario se da de alta como Pendiente para despachos y envíos.
  - Pendiente: la obra no está habilitada porque el cliente alcanzó el máximo permitido
  - Finalizada: la obra está finalizada y no cuenta en las obras habilitadas.


Se pueden agregar para cada cliente una lista de usuarios habilitados

- De cada usuario se guarda el nombre, el apellido, el dni y el correo electrónico. De esta manera tendremos en nuestra base de datos el registro de que usuarios están habilitados para hacer operaciones para que cliente.

### Reglas de negocio

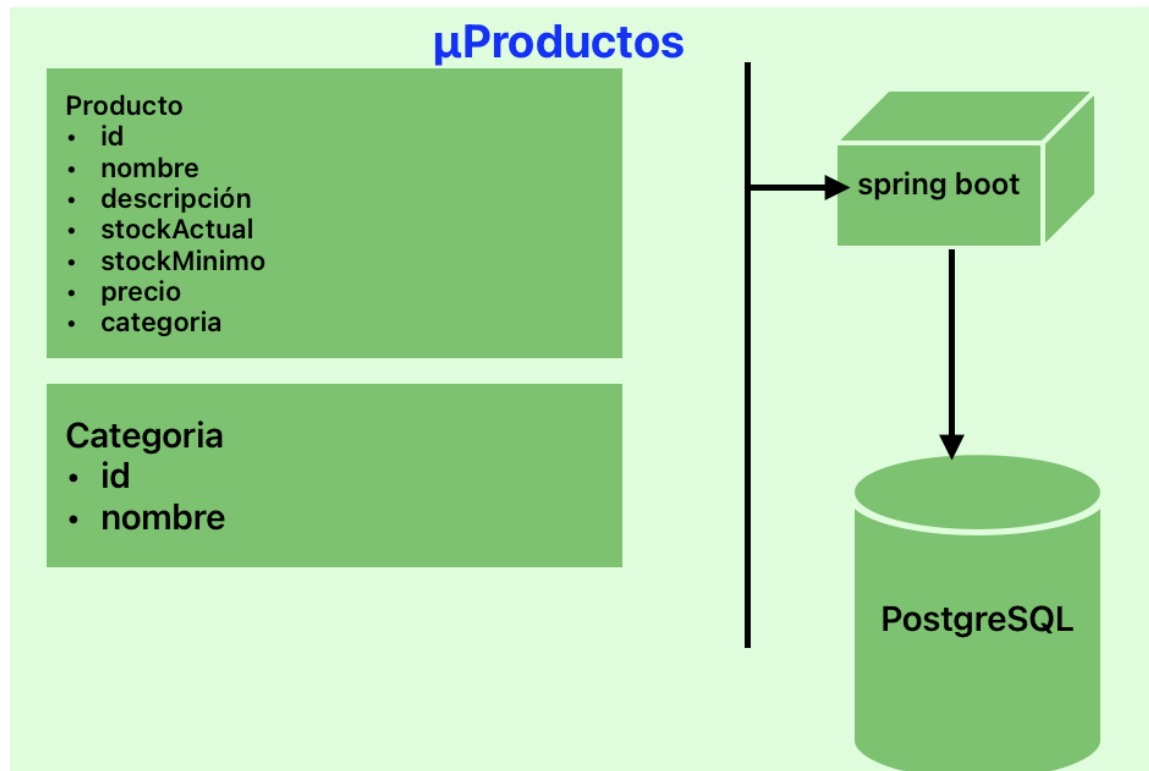
- Cuando se da de alta un cliente se debe verificar que el correo electrónico cumpla con los aspectos de formato valido usando el api de validación. 
- El máximo descubierto tiene que ser un valor mayor que cero y el valor por defecto se debe configurar en un archivo de propiedades externalizado.
- Cuando se asinga una obra a un cliente se debe verificar el estado de habilitación.
- Se puede actualizar de una obra el estado para pasarla a finalizada y se debe verificar que se pueda habilitar alguna obra que está pendiente.
- Se puede pasar una obra a estado pendiente, pero no habilita ninguna obra nueva de manera automática.
- Se puede pasar una obra a estado Habilitado y se aplicará el mismo si y solo si se cumplen las condiciones de no superar el máximo.

### Aspectos técnicos

- Deberá guardar la información en una base de datos relacional MySQL
- Configurar el plugin Jacoco y analizar cobertura 
- Crear los tests de repositorio con test containers
- Crear los tests del servicio para verificar las reglas de negocio definidas.
- Crear un test MockMvc para los controller.

## TP FINAL DAN – MS Productos



### Modelo propuesto




## Objetivos

Gestionar el alta, baja, consulta, y actualización de productos en el sistema y de las obras.

## Requisitos

- Se puede gestionar la información de un categorías a través de un endpoint rest con la información mínima obligatoria como el nombre de la categoría. 
- Se puede dar de alta un producto o eliminarlo del catálogo. Para dar de alta un producto establecer el nombre, la descripción la categoría, el stock mínimo que se desea tener del producto y el precio inicial y el **descuento promocional que se quiere aplicar sobre el producto (por defecto es 0)**. El stock inicial será 0.
- A través de un endpoint que recibe una petición PUT se recibe la información de una orden de provisión de un producto entregada por un proveedor con la información de:
  - Id de producto**
  - Cantidad de stock recibido**
  - Precio**

Con esta información recibida se debe actualizar el stock del producto y el precio del mismo.
- También debe ofrecer un endpoint PUT para actualizar el descuento promocional que quiere aplicar sobre un producto en caso de que se quiera realizar una promoción.
- Crear un servicio que consuma mensajes de una cola de mensajes JMS (usando RabbitMQ o similar) donde le llega la información de una orden de compra que es ejecutada ( este mensaje se generará en el microservicio de Pedidos) y se actualizará el stock de los productos vendidos, descontando las unidades correspondientes. 

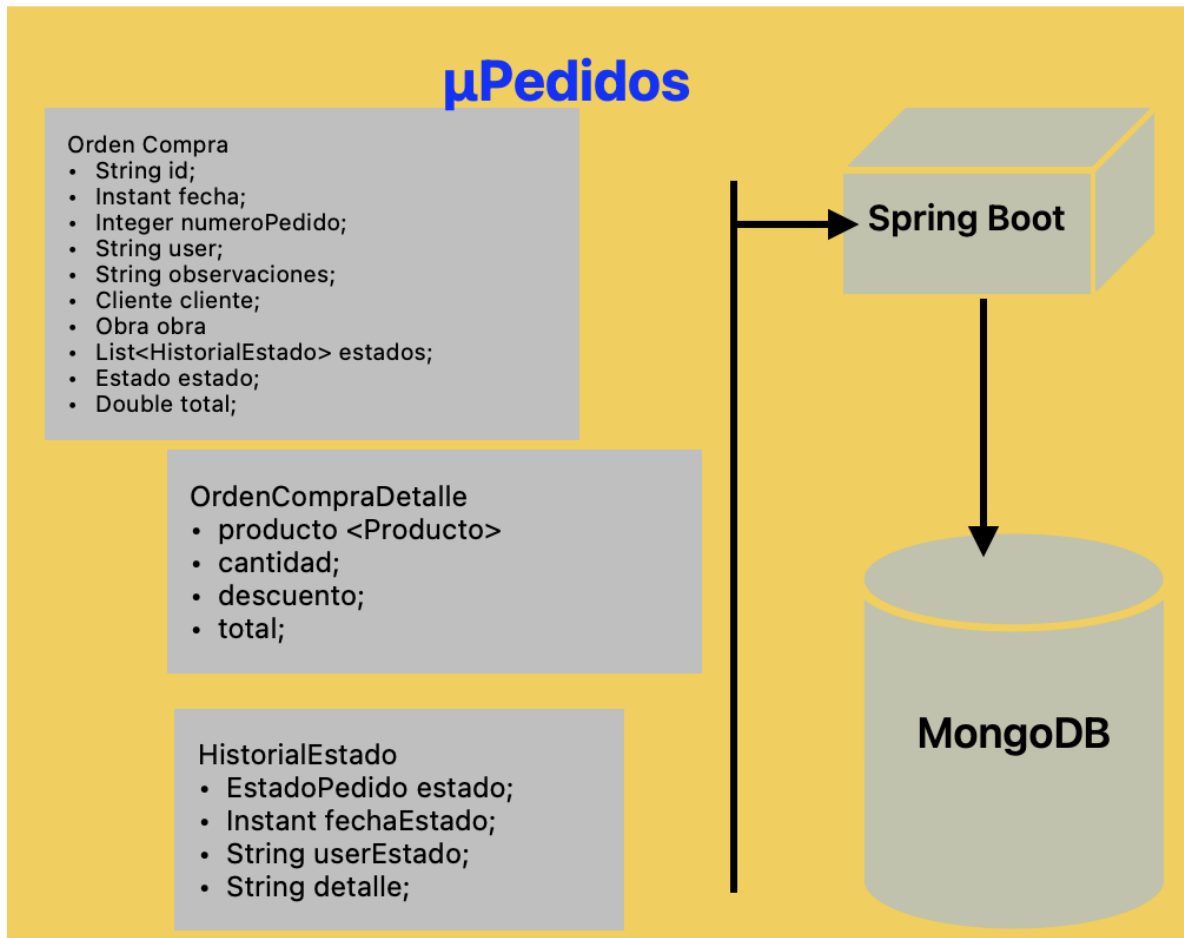
### Aspectos técnicos

- Deberá guardar la información en una base de datos relacional PostgreSQL
- Usar un servidor de mensajes asíncronos para procesar los mensajes.
- Configurar el plugin Jacoco y analizar cobertura
- Crear un test MockMvc para los controller.



## TP FINAL DAN – MS Pedidos

### Modelo propuesto



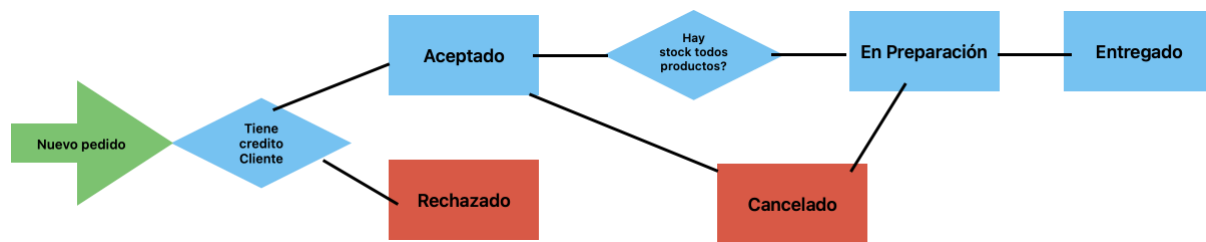
### Objetivos

Crear un microservicio spring boot que guarde información de los pedidos realizados a un sistema exponiendo un endpoint rest.

El microservicio debe exponer un endpoint rest que permita:

- POST: Crear un pedido.
- PUT: Actualizar el estado de un pedido
- GET: Consultar el estado de un pedido
- GET: Consultar los pedidos realizados por un cliente.

Un pedido puede pasar por los siguientes estados



### Descripción y reglas

- f) Creación de un pedido: Para crear un pedido hay que enviar: información del cliente, la obra a la que se debe enviar de las obras que tiene el cliente, alguna observación de ser necesario, y una lista de cada producto y la cantidad del mismo. Cuando se recibe request con el pedido de creación del pedido:
- Se le asigna un numero de pedido y una fecha de pedido (la fecha actual) y se calcula el monto total del pedido y el monto de cada línea de detalle del pedido.
  - Se verifica **sincrónicamente** a través de un endpoint rest, si el **cliente tiene saldo** suficiente para aceptar el pedido o no. Si no tiene saldo el pedido es **RECHAZADO** y el proceso termina aquí. Si tiene saldo pasa al siguiente paso que es verificar y actualizar STOCK.
  - Se actualiza **sincrónicamente** a través de un endpoint rest, el **stock para todos los productos del pedido**.. si todos los productos del pedido tienen stock suficiente y se pudo actualizar el mismo (es decir si del producto A había 10 unidades y mi pedido es de 4 unidades debo poder actualizar el stock para que indique que ahora quedan 6 unidades). Si pude actualizar el stock de todos los productos pedidos el pedido se guarda en estado **"EN\_PREPARACION"**, caso contrario queda en estado **"ACEPTADO"**
- g) **Cálculo de saldo del cliente**
- Si el cliente tiene saldo suficiente para pagar el pedido, y de ser así, se actualiza el estado a ACEPTADO, caso contrario se actualiza el estado a RECHAZADO. Si hay un error en la consulta se guarda en RECIBIDO y se actualizará más tarde. El endpoint rest retornará como resultado el pedido creado con un numero de pedido asignado automáticamente por el sistema, y una fecha.
  - Un cliente tiene saldo disponible si el monto de todos los pedidos que no fueron entregados o rechazados, más el monto del pedido actual, no superan el máximo descubierto del cliente. Ejemplo un cliente tiene un máximo descubierto de \$550 y crea el pedido 7 por \$100 y los pedidos anteriores son

Pedido	Monto	Estado
1	100	ACEPTADO
2	50	RECHAZADO
3	200	ENTREGADO
4	150	EN PREPARACION
5	100	ACEPTADO
6	150	EN PREPARACION

Sumamos el monto de los que están en estado ACEPTADO, EN\_PREPARACION y obtenemos 500. Además el nuevo pedido es por \$100 por lo que la cuenta corriente del cliente ascendería a \$600 en caso de aceptar el pedido y supera los \$550 del máximo descubierto debería rechazarse. Si a continuación carga un nuevo pedido por \$50 si se puede aceptar porque no superaría el monto máximo de descubierto.

- h) **Actualizar el estado:** el estado se puede actualizar a ENTREGADO o CANCELADO

- a. Cuando el estado se actualiza a CANCELADO se envía un mensaje JMS (RabbitMQ por ejemplo) indicando que hay que devolver el stock del pedido.

### Aspectos técnicos

- Deberá guardar la información en una base de datos relacional PostgreSQL
- Usar un servidor de mensajes asíncronos para procesar los mensajes.
- Configurar el plugin Jacoco y analizar cobertura
- Crear un test MockMvc para los controller.

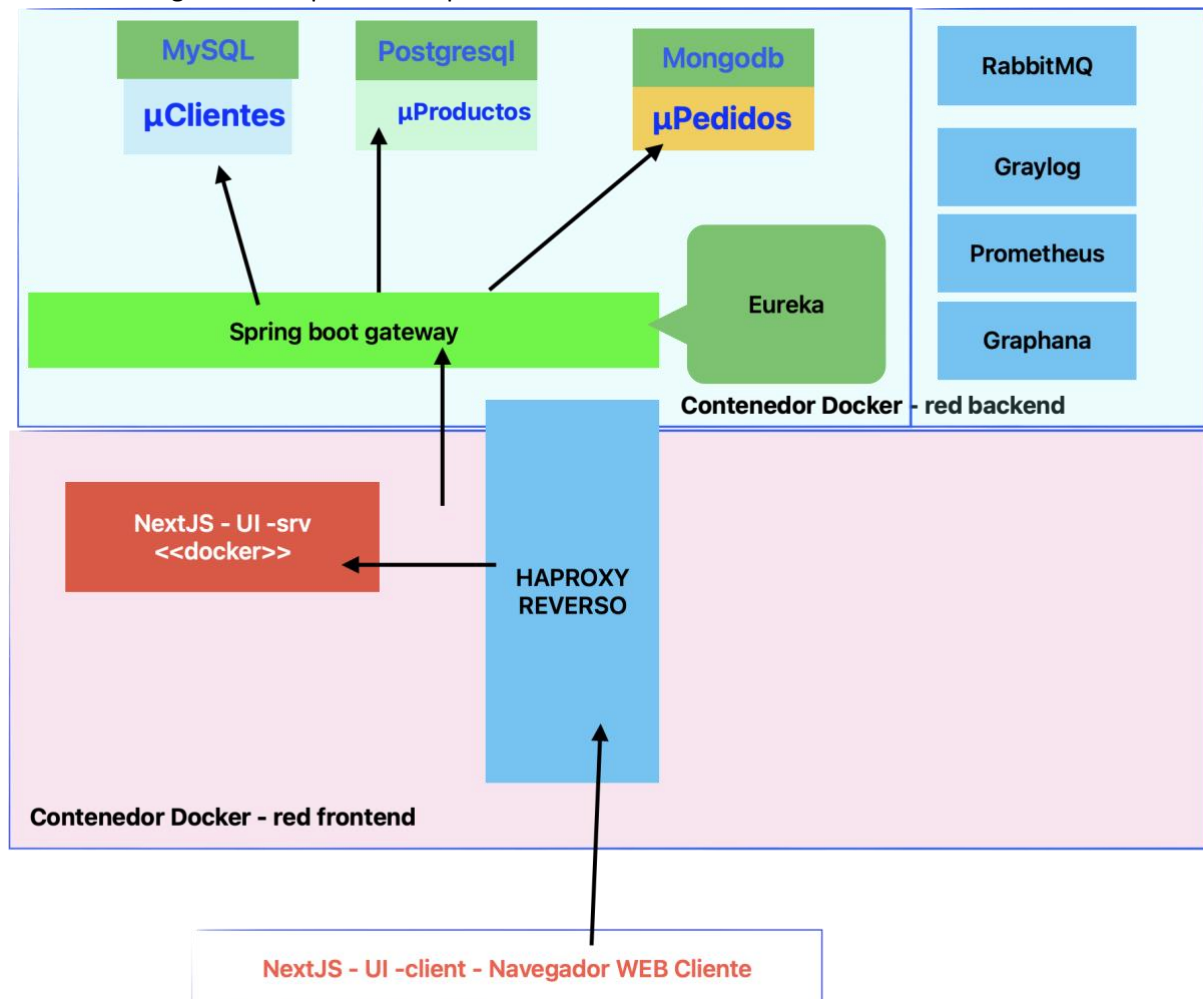


## TP FINAL DAN – UI

Esquema de infraestructura virtualizada completo del TP Final.

### Objetivos

En esta etapa se deberá crear una aplicación de UI usando React y NextJS para poder realizar desde una interface gráfica las operaciones provistas en los



microservicios anteriores.

### Operaciones requeridas

La aplicación NextJS/React permitirá:

- Dar de alta, buscar, editar y eliminar, clientes.
- Asignarle obras a los clientes.

- c) Dar de alta, buscar, editar y eliminar productos.
  - Las opciones de categoría de productos pueden estar precargadas en la base de datos no es necesario que tengan pantallas de edición.
  - La búsqueda de productos debe tener la opción de buscar por
    - Nombre
    - Código
    - Rango de precios (mínimo máximo)
    - Rango de stock (mínimo máximo)
- d) Crear un pedido.
- e) Actualizar el estado de un pedido.
- f) Buscar pedidos (por uno o varios de los siguientes criterios)
  - Por cliente
  - Por estado



**Importante: no es necesario que cree perfiles de seguridad ni que restringir el acceso a los endpoints mediante uso de token (si bien no es obligatorio puede hacerlo y sera considerado en la evaluación final).**

### Requisitos complementarios

- a) Los servicios deberán ser accedidos a través del Gateway. Entre ellos pueden comunicarse directamente, mediante el Gateway o mediante Eureka.
- b) Las peticiones desde el frontend al backend deben pasar por Haproxy y de allí al Gateway.
- c) Al menos un servicio de backend debe dejar los logs en alguna aplicación de gestión de logs por stream (Graylog o similar)
- d) Al menos uno de los servicios de backend debe publicar la información de actuador en formato prometheus y ser procesada por graphana.



Al respecto los ítems c/d no es necesario q estén “encendidos” todo el tiempo, puede configurarlo solo cuando necesite (para ahorrar recursos).

Toda la aplicación debe ejecutar en Docker, puede configurar las imágenes como desee.

### Repositorios de referencia:

- Backend: <https://github.com/dan-frsf/tp-dan-2024>
  - Microservicios
  - Gateway
  - Eureka
  - Comandos docker para levantar resto de infraestructura
- Frontend: <https://github.com/mart-domínguez/dan-ui-2024>
  - HaProxy
  - NextJS App