**CASO 01: Desarrolla un API con los siguientes requerimientos:**

**Requerimientos funcionales:**

* El sistema debe poder realizar un tipo de cambio a un monto donde se deben utilizar el monto, moneda de origen, moneda de destino, monto con el tipo de campo y el tipo de cambio.
* Debe permitir el registro, actualización y búsqueda del tipo de cambio.
* Debe tener una autenticación.
* Por cada tipo de cambio realizado, se debe registrar quien hizo la solicitud (auditoría funcional).

**Requerimientos no funcionales:**

* Lenguaje: Java.
* Framework: Spring boot.
* Utilizar programación reactiva.
* Utilizar H2 para persistencia de datos.
* La seguridad debe ser a través de JWT.
* Usar Postman o SOAPUI para el consumo de la API.
* Documentar la arquitectura de software

**Nota:**

Debe enviar el enlace del repositorio público (con todo lo requerido para desplegar la solución localmente) para ver su programación.

Solución:

https://github.com/ronel987/caso1coril.git

APIM – Azure

SAP

User CRM

Java

API -Casotipo(STS)

OAuth2

MySql

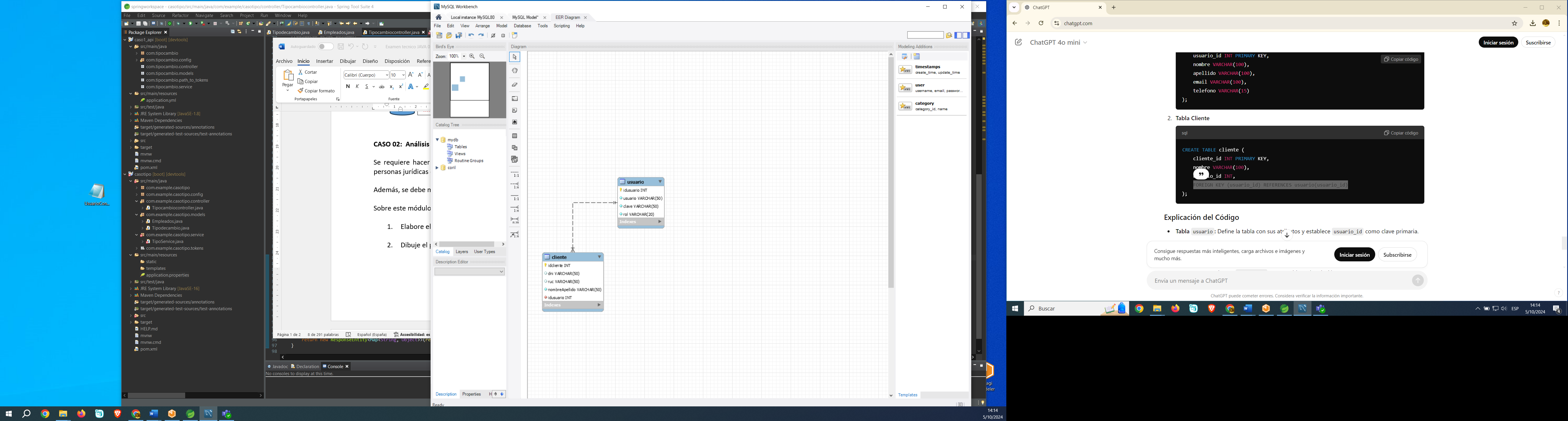
**CASO 02: Análisis y resolución de requerimientos:**

Se requiere hacer un módulo de mantenimiento para los clientes de LA EMPRESA, los clientes pueden ser personas jurídicas o naturales, siendo las naturales individuales o mancomunadas.

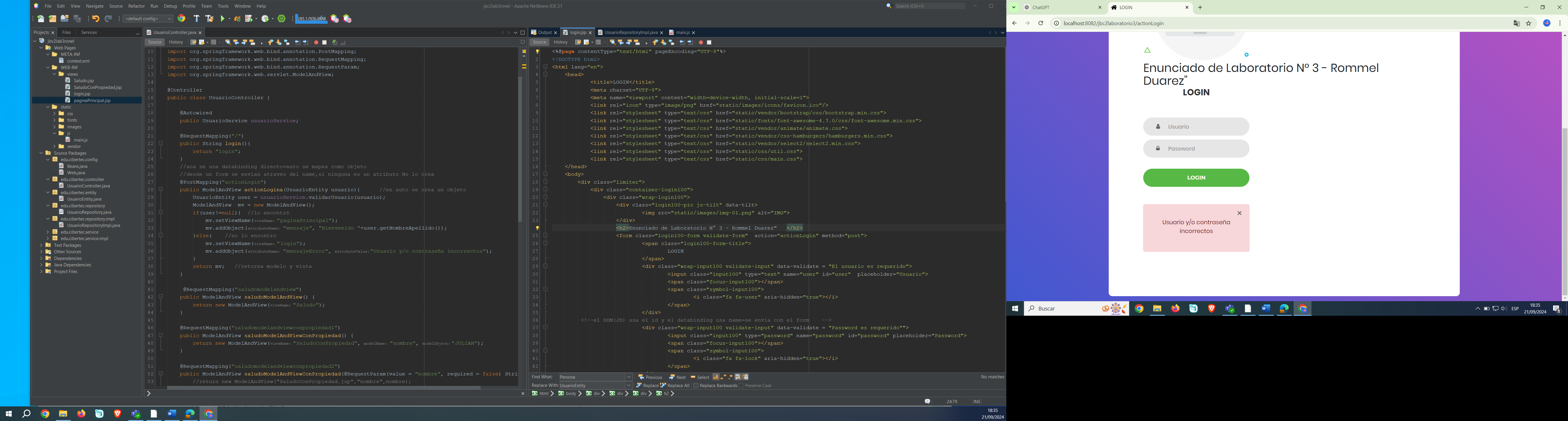
Además, se debe manejar los usuarios que estos clientes quieran usar para ingresar a la extranet.

Sobre este módulo de clientes y usuarios se requiere lo siguiente:

1. Elabore el diagrama de base de datos.



1. Dibuje el prototipo de las pantallas.



Registrar cliente:

Dni:

Ruc:

Nombre:

registrar

Actualizar cliente:

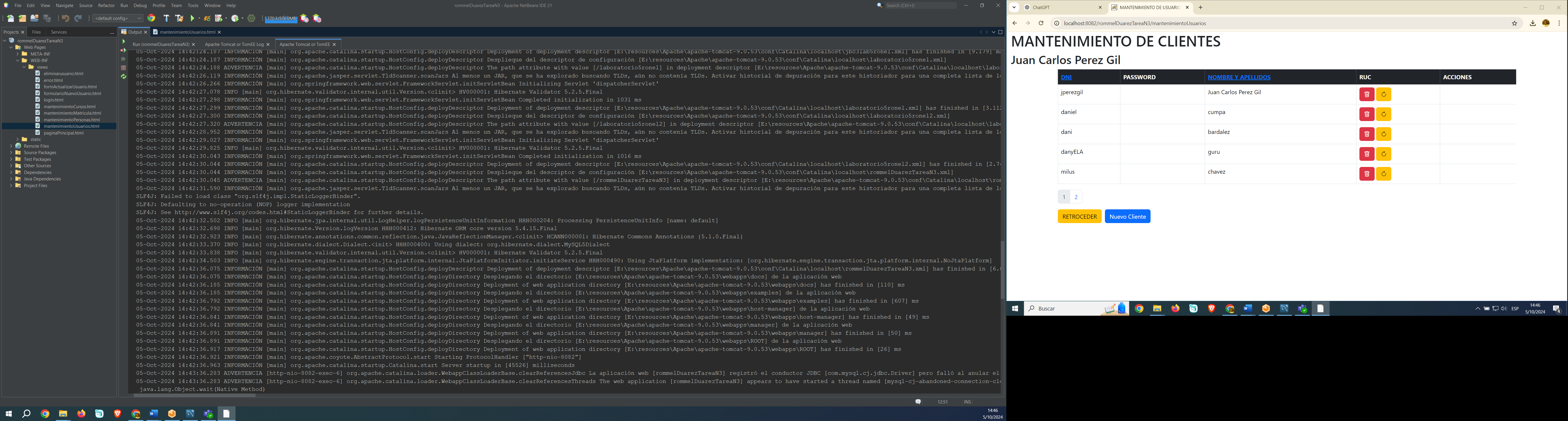
Dni:

Ruc:

Nombre:

actualizar

Listar clientes:

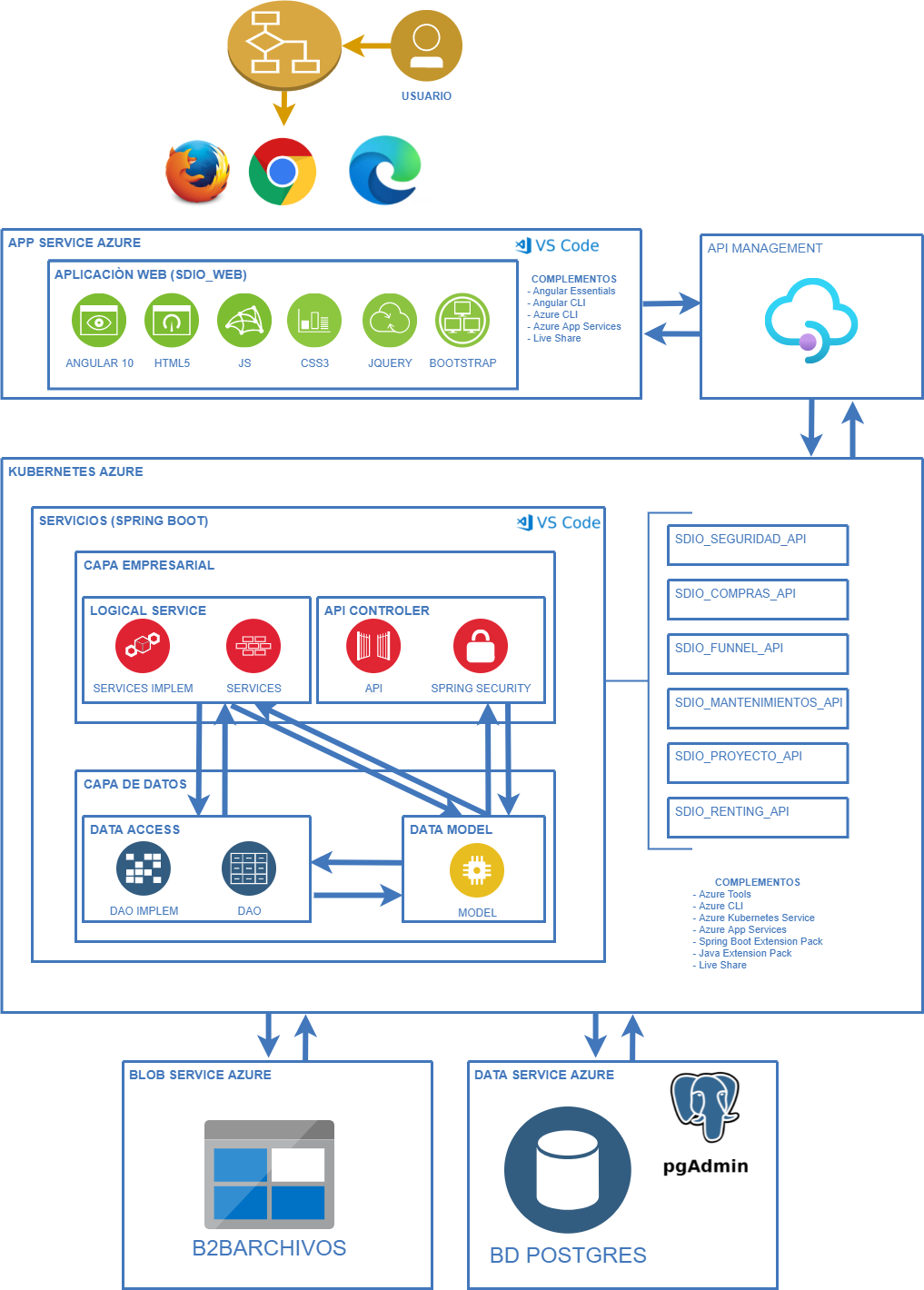


1. Diagrame la arquitectura a utilizar para la implementación: proyecto web, servicios web, escritorio, etc indicando los servidores, infraestructura necesaria.

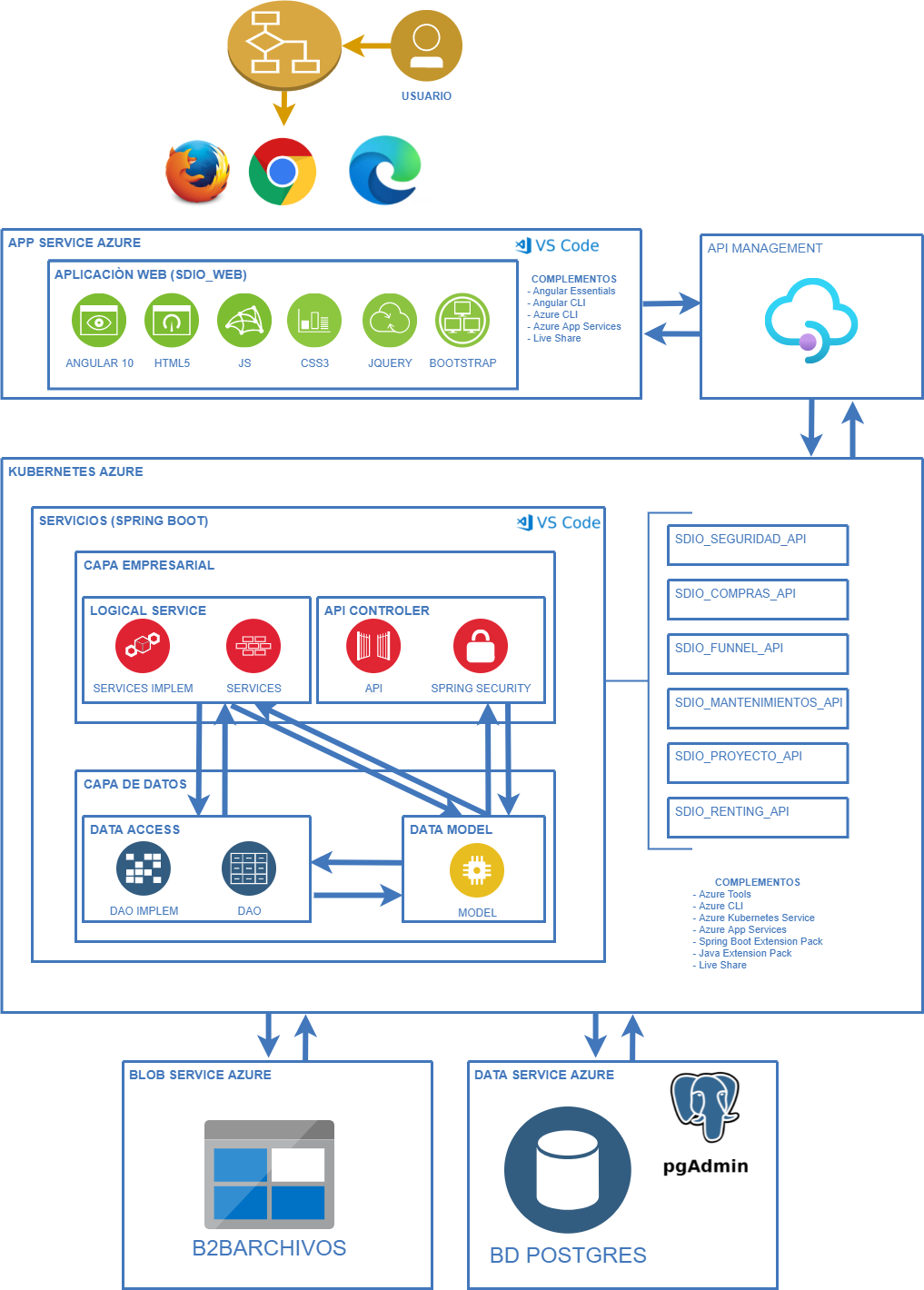
Apache Tomcat(aws)

App-web

MySql



war



1. Escriba en algún lenguaje (java, pseucodigo), indíquelo en su solución, como programaría la sección de búsqueda de los clientes (backend). Buscaría por el idcliente:

En el controlador inyecto un objeto cliente service:

@Autowired

public ClienteService clienteService;

luego cualquier método del controlador podría buscar un cliente asi (y usarlo como desee):

ClienteEntity cliente = clienteService.obtenerCliente(idcliente);

En el servicio registro ese método en la interface, asi:

public ClienteEntity obtenerCliente(Integer idcliente);

Luego en la implementación del servicio va la inyección y luego este método:

@Autowired

private ClienteRepository clienteRepository;

@Override

public ClienteEntity obtenerCliente(Integer idcliente) {

return clienteRepository.findByIdCliente(idcliente);

}

Para que funcione, en el ClienteRepository se debe extender de JpaRepository

1. Se requiere dar mantenimiento de este módulo, y le indican que un cliente no puede ingresar a la extranet. ¿Qué información pediría? Y ¿cómo lo resolvería?

Primero verificaría que el usuario y contraseña que este usando esten activos en la base de datos, sino lo están tendrían que activarse o actualizarse. También le diría que cambie de browser o limpie cache.

Segundo, verifico si esta usando bien los tokens o claves de seguridad en caso hubiese.

Tercero, verifico con Spring Actuator la salud de la app, la cantidad de recursos que tiene y la demanda

Cuarto, verifico si están llegando bien los datos del usuario y contraseña al siguiente método “actionLogin”, habiendo debug en la app en el archivo nohub del Filezilla puedo ver los valores que llegan a la variable usuario. Sino llegan adecuadamente allí está el problema

Quinto, si llegan bien los valores, verificaría la implementación del método validarUsuario() primero en el servicio y luego en el repository

@PostMapping("actionLogin")

public ModelAndView actionLogin(UsuarioEntity usuario,HttpServletRequest request){

UsuarioEntity user = usuarioService.validarUsuario(usuario);

ModelAndView mv = new ModelAndView();

if(user!=null){

mv.setViewName("paginaPrincipal");

mv.addObject("user", user);

mv.addObject("contextPath", request.getServletContext().getContextPath());

mv.addObject("method", request.getMethod());

mv.addObject("mensaje", "Bienvenido "+user.getNombreApellido());

}else{

mv.setViewName("login");

mv.addObject("mensajeError", "Usuario y/o contraseña incorrectos");

}

return mv;

}

Sexto, verifico de manera independiente en la base de datos la consulta sql para obtener el usuario