Tecnologías a utilizar:

Back

SpringBoot

SpringSecurity

Maven

Swagger

Front

Angular

Test

Junit

Mockito

Selenium

Cucumber

Jmeter

JaCoCo

Deployment

Docker(OpenShift)

Jenkins

Git

* **General**

Basandome en mi experiencia en Framework de Spring, optaría a utilizarlo como herramienta base para el desarrollo de backend.

La parte front la realizaría en Angular, que en este caso estaría integrarla al mismo proyecto back, como contenido estático, para que sea accesible desde el mismo WebLogic.

Mediante Maven y sus plugins, modular el proyecto front con la parte back. Cada modulo con su .pom correspondiente y un .pom padre. Al hacer build de back, se copiaría el contenido del directorio de “dist” de angular con las fuentes.

Con Maven también para gestionaria las dependencias, chequeo de tests, y creación del ejecutable.

Back sería una aplicación con las siguientes capas: Controlador, Servicio, Modelo, Repositorio,Seguridad.

* Los controladores serian Rest.
* Los servicios seguirían el patrón DTO respecto al modelo de datos obtenidos desde los repositorios.
* Los repositorios seguirían el patrón DAO, y no consultarían los datos a DDBB si no que al servicio de Facturador.
* Seguridad por medio de SpringSegurity, a la hora de autentificación desde la web, que se haría según el sistema que esté implementado por la empresa o hacer una propia como token o por BBDD.

Desconozco el método de comunicación con el servicio Facturador, en el caso de que sea una interfaz REST, Spring dispone RestTemplate mediante el cual permitiría hacer cualquier tipo de petición hacia este servicio.

No tengo mucha experiencia con la interfaz SOAP, pero Spring dispone de librería JAXB que permite trabajar con ficheros XML, y libreria WS que permitiría la comunicación con el servicio CRM, para hacer cualquier tipo de peticiones.

Comunicación entre el fron y back se haria por medio de Rest. Con el cliente http de angular y los endpoints de los controladores.

* **Test**

Con las herramientas Junit y Mock podemos realizar las pruebas unitarias como de integración, que en este caso seria contra el servicio Facturador o una BBDD embebida.

Al integrar Swagger dentro del proyecto, compruebo cada uno de los endpoints.

La parte web creada con jsp en algún proyecto anterior, yo había testeado con Selenium y Cucumber, era una solución bastante practica, porque estos tests los hacías dentro del código, y no por medio de alguna aplicación externa, lo único que no pudimos integrarlos dentro de los test Junit, porque para que los escenarios se ejecuten el proyecto debe de estar en ejecución, y los test Junit pasan antes del arranque, así que, los test de la web los teniamos en otro proyecto a parte, en la misma rama del proyecto git.

Conozco Jmeter como una aplicación stress test, pero en el proyecto que estuve no estaba implementada, lo único que puede hacer antes de que me cambiaran de proyecto, era integrar el fichero de configuración de Jmeter al git y que cada usuario lo ejecutara en su maquina local.

En caso de que en la política de empresa exista cobertura de pruebas de código, incluir Jacoco. Aunque solo he tenido experiencia con ello unos meses.

* **Despliegue**

Para el desarrollo, una vez echo el build , obtendría el war del proyecto, buscaría una imagen con WebLogic en Docker, lo arrancaría y desplegaría el war generado. O creando un pipeline de Jenkins para que lo haga una ve echo el commit en rama.