TABLA DE CONTENIDO

[1. Descripción 3](#_Toc6601209)

[2. Propósito 3](#_Toc6601210)

[3. Arquitectura 3](#_Toc6601211)

[a. Front-end (Angular7) 3](#_Toc6601212)

[b. Backend(Spring Java y Mysql) 3](#_Toc6601213)

[c. Rest(Maven) 3](#_Toc6601214)

[d. Deploy(Tomcat, AWS,Maven) 3](#_Toc6601215)

[4. Estructura 4](#_Toc6601216)

[5. Tecnología a utilizada 4](#_Toc6601217)

[a. Capa de presentación (Front End) 4](#_Toc6601218)

[i. HTML5 4](#_Toc6601219)

[ii. Bootstrap 4](#_Toc6601220)

[b. Controlador (Front End) 4](#_Toc6601221)

[i. AngularJS version 7 4](#_Toc6601222)

[c. Servicio Rest (Back – End) 5](#_Toc6601223)

[i. Spring Boot Framework 5](#_Toc6601224)

[d. Pruebas Unitarias 5](#_Toc6601225)

[i. JsUnit integrado en el IDE de desarrollo se generaban los cocos para validar las acciones. 5](#_Toc6601226)

[e. Automatización de construcción de código 5](#_Toc6601227)

[i. Maven integrado en el IDE desarrollo realizaba todo(depuración, compilación y ejecución de pruebas unitarias) de una vez cada vez se ejecute desde el IDE de desarrollo la aplicación. 5](#_Toc6601228)

[f. Pre carga de información 5](#_Toc6601229)

[i. script de Inserción de datos previo 5](#_Toc6601230)

[g. Base de datos 5](#_Toc6601231)

[i. Mysql ENGINE = MEMORY 5](#_Toc6601232)

[h. Deploy Servidor de aplicaciones 5](#_Toc6601233)

[i. Apache Tomcat 5](#_Toc6601234)

[6. Publicacion 6](#_Toc6601235)

[a. GitHub 6](#_Toc6601236)

[b. Back-end 6](#_Toc6601237)

[i. War 6](#_Toc6601238)

[c. Front-End 6](#_Toc6601239)

[i. Dist 6](#_Toc6601240)

[7. Extras 6](#_Toc6601241)

[a. AWS 6](#_Toc6601242)

# Descripción

Realizar una prueba técnica que involucre las herramientas indicadas en el desarrollo Full-Stack de aplicaciones, para la siguiente prueba adjunta en la imagen siguiente:



# Propósito

El propósito de este documento es realizar una prueba que servirá como información técnica para demostrar conocimientos adquiridos y aplicados para el proceso de selección.

# Arquitectura

## Front-end (Angular7)

Desarrolle el front-end de las 3 paginas solicitadas creando un componente y servicio por cada elemento del CRUD solicitado, dando total de 4 componentes cada uno con su servicio, se utilizó HTML5, CSS, TYPESCRIPT basado en Framework de angular con modelo MVC.

## Backend(Spring Java y Mysql)

Utilizando el framework springboot organice la construcción de clases y paquetes según el modelo MVC generando los servicios web para enviar los objetos de datos desde base de datos por medio de Java.

## Rest(Maven)

Con las anotaciones y mapeos de Java para las peticiones desde Front-end se crearon y configuraron los servicios de inserción, consulta, selección y borrado (CRUD).

## Deploy(Tomcat, AWS,Maven)

Por medio del servidores de aplicaciones he generado el WAR para el back-end y los archivos compilados de angular de la carpeta DIST para el Front-end para poder cargar la aplicación.

# Estructura

Visualizando el modelo de base de datos para las relaciones entre tablas, se crearon 4 tablas consumos, tarjetas, clientes y asesores.

# Tecnología a utilizada

## Capa de presentación (Front-end)

### HTML5

Cree en total 13 páginas HTML que son el índex y el resto de las 12 paginas son para cada componente y correspondiente acción del CRUD.

### Bootstrap

Ultima version.

## Controlador (Front-end)

### AngularJS version 7

Lista de componentes y servicios desarrollados.

* ng g c Cliente/listarCliente
* ng g c Cliente/addCliente
* ng g c Cliente/editCliente
* ng g c Tarjeta/listarTarjeta
* ng g c Tarjeta/addTarjeta
* ng g c Tarjeta/editTarjeta
* ng g c Consumo/listarConsumo
* ng g c Consumo/addConsumo
* ng g c Consumo/editConsumo
* ng g c Asesor/listarAsesor
* ng g c Asesor/addAsesor
* ng g c Asesor/editAsesor
* ng g s Service/serviceCliente
* ng g s Service/serviceTarjeta
* ng g s Service/serviceConsumo
* ng g s Service/serviceAsesor

## Servicio Rest (Back–end)

### Spring Boot Framework

## Pruebas Unitarias

### JsUnit

### Integrado en el IDE de desarrollo se generaban los cocos para validar las acciones.

## Automatización de construcción de código

Maven integrado en el IDE desarrollo realizaba todo (depuración, compilación y ejecución de pruebas unitarias) de una vez cada vez se ejecute desde el IDE de desarrollo la aplicación.

## Pre carga de información

### script de Inserción de datos previo

## Base de datos

### Mysql

* CREATE TABLE `prueba01db`.`clientes` ( `id` INT NOT NULL AUTO\_INCREMENT , `nombre` VARCHAR(50) NOT NULL , `direccion` VARCHAR(100) NOT NULL , `ciudad` VARCHAR(30) NOT NULL , `telefono` VARCHAR(20) NOT NULL , PRIMARY KEY (`id`)) ENGINE = MEMORY;
* CREATE TABLE `prueba01db`.`tarjetas` ( `cliente\_id` INT NOT NULL , `id` INT(16) NOT NULL , `ccv` INT(4) NOT NULL , `tipo` VARCHAR(50) NOT NULL ) ENGINE = MEMORY;
* [ALTER](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) `tarjetas` ADD CONSTRAINT `fk\_cliente` FOREIGN KEY (`cliente\_id`) REFERENCES `clientes`(`id`) ON [DELETE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/delete.html) cascade ON [UPDATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/update.html) cascade;
* CREATE TABLE `prueba01db`.`consumos` ( `tarjeta\_id` INT NOT NULL , `fecha` DATETIME NOT NULL , `descripcion` VARCHAR(100) NOT NULL , `monto` DECIMAL(12,2) NOT NULL ) ENGINE = MEMORY;
* [ALTER](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) `consumos` ADD CONSTRAINT `fk\_tarjeta\_id` FOREIGN KEY (`tarjeta\_id`) REFERENCES `tarjetas`(`id`) ON [DELETE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/delete.html) CASCADE ON [UPDATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/update.html) CASCADE;

## Deploy Servidor de aplicaciones

### Apache Tomcat

* Desplegar el archivo WAR de la ruta 6.b.i WAR de este documento en el servidor apache Tomcat y nombrar el despliegue con el mismo nombre del archivo.
* Desplegar todo el contenido de archivos, páginas y scripts de la ruta 6.b.i. en un solo despliegue.
* Configurar el servidor de aplicaciones por el puerto 8080.
* Ejecutar el servidor de aplicaciones al puerto 8080 seguido del nombre del despliegue realizado en 6.b.i. Así:
  + http://{servidordeaplicaciones}:8080/{nombredeldespligue de 6.b.i. }

# Publicacion

## GitHub

Todo el Proyecto código fuente, documentación y despliegues.

<https://github.com/jthemaster/pruebatecnica>

## Back-end

Código fuente completo.

<https://github.com/jthemaster/pruebatecnica/tree/master/JavaSpringServer>

### War

Archivo para desplegar en producción. <https://github.com/jthemaster/pruebatecnica/blob/master/Deploy/prueba02service.war>

## Front-end

Código fuente completo.

<https://github.com/jthemaster/pruebatecnica/tree/master/src>

### Dist

Todo el código Fuente para desplegar en producción.

<https://github.com/jthemaster/pruebatecnica/tree/master/Deploy/dist/prueba01>

# Extras

## AWS

Configuración y subida del Servidor Apache Tomcat sobre instancias de Amazon Elastic Compute Cloud (EC2) instalando Java.

A continuación, dejo el link de publicación.

<http://pruebatecnicaibm.ddns.net>

A continuación, dejo el link de publicación alternativo.

<http://pruebatecnicaibm.ddns.net:8080/prueba01/>