Creado con Spring Tool Suite 4 Version: 4.1.1.RELEASE

API REST

Workspace y repo local: C://Users//guido//workspacedev//basketbackend

Repositorio en https://gitlab.com/guidobertinat/basketbackend

Configuracion local/produccion de PATH para subir fotos

Los controllers deben tener el @CrossOrigin(origins = "\*", maxAge = 3600), para aceptar solicitudes desde JS, desde cualquier sitio. Ver si cambiar como bestpractice.

El path tiene que estar abierto en SecurityConfig

PlayerController:

**private** **static** String *UPLOADED\_FOLDER* = "C://Users//guido//workspacedev//basketbackend//src//main//webapp//";

Con esta ruta (webapp) impacta en dominio.com:8081/basketbackend

(se instala en tomcat)

En /opt/apache-tomcat-8.5.37/webapps/, al mover el .war en esta carpeta de descomprime y deploya solo, luego reiniciar servidor.

En carpeta /opt/tomcat/bin :

Sh shutdown.sh (apaga servidor tomcat)

Sh startup.sh (levanta servidor tomcat)

Ps –aux para ver los procesos, kill -9 14103(el pid) para matar proceso si esta colgado.

Esto puede ir como standalone y correr como un java con java -jar prueba.jar

como un servicio. En este caso no hace falta extender de ninguna clase la app principal. (En desarrollo como el Spring Tools incluye un server embebido en la app, no hace falta y no extiende de nada tampoco)

O puede ir como una aplicación a ejecutar en un tomcat instalado, como esta deployado en este proyecto.

Generando un .war, extendiendo la clase app (principal) de SpringBootServletInitializer. Como tenemos en el workspace del proyecto productivo.

public class App extends SpringBootServletInitializer {

El archivo se genera con botón derecho 🡪 export 🡪 WEB/War File

C://Users//guido//workspaceprod//basketbackend

El archivo application.properties tiene distintas configuraciones del proyecto, timeout de token, puerto, claves, etc.

En la API java utilizo otro workspace de produccion para compilar con los archivos modificados, trabajo en desarrollo y luego muevo cambios (los bajo desde gitlab o manual). Antes de hacer pull para bajar los datos que están subidos en el repositorio, tengo que hacer un stash para guardar los cambios en los archivos productivos, una vez hecho el pull, hago el stash pop, luego compilo y puedo subir.

Al revés de la parte de react que el repositorio master es el productivo.

Utilizo MAVEN, con la configuración del proyecto en el pom.xml

Para upload y read de archivos (fotos en server) hay que darle permisos al usuario y crear el directorio.

chown -R www-data:www-data /var/www/html

chmod -R 0550 /var/www/html

Path es la ruta a un objeto del Sistema, es una nueva forma de trabajar files.

\*\*\*Resources placed in src/main/resources/public will be served from the root of your application. For example src/main/resources/public/hello.jpg would be served from <http://localhost:8080/hello.jpg>

Log de excepciones no controladas, como los System.***out***.println, van en server/logs/catalina.out

Si tenemos un servidor de mail podemos enviar el stack trace desde el bloque catch, si manejamos bien las excepciones.

La otra opción es utilizar log4j para escribir en otro archivo con el nivel que queramos de prioridad.

try {

// Código que declara lanzar excepciónes

} catch(Exception ex) {

logging.log("Se ha producido el siguiente error: " + ex.getMessage());

logging.log("Se continua la ejecución");

}

Files.createFile(file.toPath(), attr);

Files.setPosixFilePermissions(file.toPath(), defaultPosixPermissions);

Crea archivo, luego setea permisos.

\*Repositorio Java local Producción: Dejo commiteados en local en el entorno productivo los archivos configurados para el compilado y deploy, asi puedo ir mergeando con lo que voy subiendo de la rama desarrollo. Si no esta comiteado no lo mergea.

¿Autorización JWT como funciona?

Login: https://github.com/AgileIntelligence/AgileIntPPMTool/tree/branch66

Tabla USERS, en mysql AWS no se puede usar palabras reservadas como USER, para tablas.

Organización GIT para FRONTEND

Existen dos ramas, master que será igual a producción y dev que será la de desarrollo pero a su vez cumplirá la función de reléase.

Es decir, lo cambios de los desarrolladores se realizaran sobre ella (commit, push, pull), y una vez llegado a un punto, iremos a master y haremos un merge de todos los cambios realizados tanto por el desarrollador 1 y el desarrollador 2.

Los desarrolladores deberán tener el branch DEV como referencia, ya que no contendrá los archivos configurados productivos, como los tiene master.

Una vez integrados en master, habrá que transpilar y subir al servidor.

Utilizo node.js (este se utiliza para crear y gestionar la aplicación en react, tanto para transpilar como para generar deployable) la aplicación se crea con create-react-app

Una vez en la dirección del repositorio se inicia con npm start

Deploy de React en desarrollo y producción

Se utilizan dos ramas, una master que tiene la versión productiva para compilar y es subida a github, y otra rama dev en local, con la configuración para desarrollo.

Trabajo en dev, y hago merge en master antes de push a github.

En axiosUrl:

const clienteAxios = 'http://18.220.188.121:8080/basketbackend'

En App.js

        <BrowserRouter basename="/hispano" >

En package.json

 "homepage": "http://http://18.220.188.121/hispano/",

Tambien reemplazo

  "browserslist": {

    "production": [

      ">0.2%",

      "not dead",

      "not op\_mini all"

    ],

    "development": [

      "last 1 chrome version",

      "last 1 firefox version",

      "last 1 safari version"

    ]

Por

  "browserslist":   []

En store.js (redux)

/\*

if (window.navigator.userAgent.includes("Chrome")) {

  store = createStore(

    rootReducer,

    initalState,

    compose(

      applyMiddleware(...middleware),

      window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_\_ &&

        window.\_\_REDUX\_DEVTOOLS\_EXTENSION\_\_()

    )

  );

} else {

  store = createStore(

    rootReducer,

    initalState,

    compose(applyMiddleware(...middleware))

  );

}\*/

store = createStore(

  rootReducer,

  initalState,

  compose(applyMiddleware(...middleware))

);

Para desarrollo se utiliza la parte comentada, que es para poder usar la Redux Dev Tool y ver los estados.

Crear serverURL, agregar en TheTeam

import serverURL from '../../config/**serverURL'**;

const serverURL = 'http://18.220.188.121/'

//const servidorURL = 'http://18.220.188.121/'

export default serverURL;

Para que funcione el route JS en subdirectorio (y en root también puede ser)

Generar archivo .htaccess en ruta del server /var/www/html/hispano (para ver archivos ocultos en Linux ls –a)

RewriteEngine On

RewriteCond %{DOCUMENT\_ROOT}%{REQUEST\_URI} -f [OR]

RewriteCond %{DOCUMENT\_ROOT}%{REQUEST\_URI} -d

RewriteRule ^ - [L]

# Fallback all other routes to index.html

RewriteRule ^ /hispano/index.html [L]

Luego ejecutar: sudo a2enmod rewrite

Y reiniciar con: service apache2 restart

Luego de todo esto:

npm run build

Copiar datos de carpeta generada build (en directorio del repo) al servidor (en este caso el apache en <http://18.220.188.121>) (no en tomcat, esto es estatico, sin java)

Para instalar otra computadora, hay que instalar nodejs, utilizar la consola (prompt) hasta el directorio donde tenemos el proyecto. Esto lo hacemos con un git clone.

Luego hacemos un npm install, y luego npm start.