Instalación TFG PRODUCCIÓN

**5.3.3 Instalación Docker**

# Clona o descarga el proyecto

# Asegúrate de tener Docker y Docker Compose instalados

# Verificar instalación

docker --version

docker-compose --version

### 2. **Despliegue Automático (5-10 minutos)**

# Dar permisos de ejecución al script

chmod +x deploy.sh

# Ejecutar el despliegue

./deploy.sh

### 3. **Acceso a la Aplicación**

* **Frontend:** <http://localhost>
* **Backend API:** <http://localhost:8080>
* **Swagger UI:** [http://localhost:8080/swagger-ui/index.html](http://localhost:8080/index.html)

### 4. **Credenciales por Defecto**

* **Admin:** [admin@gmail.com](mailto:admin@gmail.com) / admin
* **Usuario:** [user@gmail.com](mailto:user@gmail.com) / user

**Instalación para producción en sevidor linux**

Dependiendo de tu distribución linux pueden cambiar un poco los pasos pero son casi siempre muy parecidos. Yo he usado Ubuntu-server por eso utilizo el gestor de paquetes apt. También lo he probado para la distribución amazon linux y cambian varias cosas el gestor de paquetes que es yum y como configuras los puertos 80 y 443 que es desde /etc/httpd/conf.

**como instalar los requisitos de software**

apache:

sudo apt update

sudo apt install -y httpd

sudo systemctl start httpd

sudo systemctl enable httpd

sudo systemctl status httpd

git:

sudo apt install git

mysql:

sudo apt install mysql-community-server

sudo systemctl start mysqld

sudo systemctl enable mysqld

-- pon este comando para ver la contraseña temporal

sudo grep 'temporary password' /var/log/mysqld.log

inicia sesión con mysql -u root -p y la contraseña

--cambia la contraseña con ALTER USER 'root'@'localhost' IDENTIFIED BY 'Abc1234@'; (mínimo 8 caracteres, con letras, números y caracteres especiales)

NodeJS:

# Descarga e instala nvm:

curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.2/install.sh | bash

# en lugar de reiniciar la shell

\. "$HOME/.nvm/nvm.sh"

# Descarga e instala Node.js:

nvm install 22

JDK 21:

wget <https://download.java.net/java/GA/jdk21.0.1/415e3f918a1f4062a0074a2794853d0d/12/GPL/openjdk-21.0.1_linux-x64_bin.tar.gz>

mkdir -p ~/java

tar -xzf openjdk-21.0.1\_linux-x64\_bin.tar.gz -C ~/java

-- Editar el archivo .bashrc

nano ~/.bashrc

-- Configurar JAVA\_HOME y PATH **al final del archivo**

export JAVA\_HOME=~/java/jdk-21.0.1

export PATH=$JAVA\_HOME/bin:$PATH

source ~/.bashrc

Al poner java -version deberia salirte esto

openjdk version "21.0.1" 2023-10-17

OpenJDK Runtime Environment (build 21.0.1+12-29)

OpenJDK 64-Bit Server VM (build 21.0.1+12-29, mixed mode, sharing)

**5.3.2 Instalación paso a paso**

**1. Configuración de la base de datos:**

*-- Crear base de datos*

CREATE DATABASE reparaciones CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci;

*-- Crear usuario*

CREATE USER 'repair\_user'@'localhost' IDENTIFIED BY 'Abc1234@';

*-- Otorgar permisos*

GRANT ALL PRIVILEGES ON reparaciones.\* TO 'repair\_user'@'localhost';

FLUSH PRIVILEGES;

**2. Configuración del backend:**

*# Clonar repositorio*

git clone <https://github.com/IvanCdeveloper/TFG-backend>

cd TFG-backend

*# Editar configuración de base de datos*

nano src/main/resources/application.properties

Configuración de application.properties:

# Configuración del servidor

spring.datasource.username=repair\_user

spring.datasource.password=Abc1234@

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# Configuración JPA

spring.jpa.show-sql=false

Lo demás dejalo igual

**3. Compilación y ejecución del backend:**

*# Compilar proyecto*

mvn clean compile

*# Crear JAR ejecutable*

mvn package

*# Ejecutar aplicación*

*# /etc/systemd/system/tienda-reparaciones.service*

*nano /etc/systemd/system/tienda-reparaciones.service*

[Unit]

Description=Tienda Reparaciones Backend

After=network.target

[Service]

Type=simple

User=repair-app

# Sustituir usuario por el usuario de tu servidor dónde tengas el proyecto

WorkingDirectory=/home/usuario/tienda-reparaciones

ExecStart=/usr/bin/java -jar tienda-reparaciones-0.0.1-SNAPSHOT.jar

Restart=always

RestartSec=10

[Install]

WantedBy=multi-user.target

# Ejecuta el servicio

sudo systemctl daemon-reload

sudo systemctl start tienda-reparaciones.service

# Comprueba el estado

sudo systemctl status tienda-reparaciones.service

**4. Configuración del frontend:**

# Movernos a /var/www/html

cd /var/www/html

# Clonar el frontend

*# Navegar al directorio frontend*

cd ../../frontend/tienda-reparaciones

*# Instalar dependencias*

npm install

*# Configurar environment*

cp src/app/environments/environment.example.ts src/app/environments/environment.ts

*# Editar configuración del environment*

nano src/app/environments/environment.ts

Configuración de environment.ts:

export const environment = {

  production: true,

  companyName: 'Reparaciones',

  companyName2: 'App',

  companySlogan: 'Reparamos cualquier dispositivo móvil',

  baseUrl: 'https://ivancuarteros.com/api'

};

**5. Compilación y ejecución del frontend:**

*# Desarrollo*

ng serve

*# Producción*

ng build  --base-href /tienda-reparaciones/

# Esto generará una carpeta en dist/tienda-reparaciones/browser con toda la página web de

# angular.

# Hay que mover todos los archivos de la carpeta /browser a la carpeta /tienda-reparaciones

cd /dist/tienda-reparaciones

shopt -s dotglob nullglob

mv browser/\* .

**3. Configuración de proxy reverso (Apache puerto 80):**

<VirtualHost \*:80>

    ServerName ivancuarteros.com

    ServerAlias www.ivancuarteros.com

</VirtualHost>

**Apache puerto 443 (https):**

#  El ServerName y el server alias cambiarlo por vuestro dominio

<IfModule mod\_ssl.c>

<VirtualHost \*:443>

   ServerName ivancuarteros.com

    ServerAlias www.ivancuarteros.com

    DocumentRoot /var/www/html

    # SSL Configuration

    SSLEngine On

    SSLCertificateFile /etc/letsencrypt/live/ivancuarteros.com/fullchain.pem

    SSLCertificateKeyFile /etc/letsencrypt/live/ivancuarteros.com/privkey.pem

    # SSL Security

    SSLProtocol all -SSLv2 -SSLv3 -TLSv1 -TLSv1.1

    SSLCipherSuite ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384

    SSLHonorCipherOrder off

    SSLSessionTickets off

    # HSTS

    Header always set Strict-Transport-Security "max-age=63072000; includeSubDomains; preload"

    # Configuración para Angular (SPA)

    <Directory /var/www/html>

        Options Indexes FollowSymLinks

        AllowOverride All

        Require all granted

        # Configuración para Angular Router

        RewriteEngine On

        RewriteBase /

        RewriteRule ^index\.html$ - [L]

        RewriteCond %{REQUEST\_FILENAME} !-f

        RewriteCond %{REQUEST\_FILENAME} !-d

        RewriteRule . /index.html [L]

    </Directory>

    # Proxy para API Backend con SSL

    SSLProxyEngine On

    SSLProxyVerify none

    SSLProxyCheckPeerCN off

    SSLProxyCheckPeerName off

    SSLProxyCheckPeerExpire off

    ProxyPreserveHost On

    ProxyPass /api/ http://localhost:8080/api/

    ProxyPassReverse /api/ http://localhost:8080/api/

</VirtualHost>

</IfModule>

# Redirección HTTP a HTTPS

<VirtualHost \*:80>

    ServerName ivancuarteros.com

    ServerAlias www.ivancuarteros.com

    RewriteEngine On

    RewriteCond %{HTTPS} off

    RewriteRule ^(.\*)$ https://%{HTTP\_HOST}%{REQUEST\_URI} [R=301,L]

</VirtualHost>

**Módulos de apache que son necesarios activar:**

# Habilitar módulos

sudo a2enmod rewrite

sudo a2enmod ssl

sudo a2enmod headers

sudo a2enmod expires

sudo a2enmod deflate

sudo a2enmod proxy

sudo a2enmod proxy\_http

sudo a2enmod proxy\_balancer

sudo a2enmod lbmethod\_byrequests

# Reiniciar Apache para aplicar cambios

sudo systemctl restart apache2