Fraxen - Website documentation

Basically this is how It all started:

1. Which content is labeled as “absolutely crucial” on the future website?

2. Which content is labeled as “nice to have”?

3. Which actions would you like the website visitors to perform?

4. Other preferences regarding the website (theme, font ..)

Answerer:

- “ Cine suntem si care este value proposition ”

- “ Profilele ”

- “ Contact , understand value proposition “

- “ Color purple “

Our social image is also important, so I’ve managed to create an email [admin@fraxen.eu](mailto:admin@fraxen.eu) for us to declare as an official email, aswell as a linkedin company and a linkedin group for Fraxen / Fraxen – Group.

After designing the interface of the presentation page, I went ahead and hosted the POSTGRESQL database on a virtual machine. With the database up, I linked my laptop to it and I started to create the place where we would have the data for our executives stored. Here’s the query I used to create the table:

-- Main executives table

CREATE TABLE executives (

id SERIAL PRIMARY KEY,

gender VARCHAR(20) NOT NULL,

title VARCHAR(255) NOT NULL,

experience VARCHAR(100) NOT NULL,

sector\_focus TEXT NOT NULL,

location VARCHAR(255) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Executive highlights/positions table (one-to-many relationship)

CREATE TABLE executive\_highlights (

id SERIAL PRIMARY KEY,

executive\_id INTEGER NOT NULL,

position\_title VARCHAR(255) NOT NULL,

company\_description VARCHAR(255),

details TEXT NOT NULL,

display\_order INTEGER NOT NULL DEFAULT 0,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (executive\_id) REFERENCES executives(id) ON DELETE CASCADE

);

-- Core strengths table (one-to-many relationship)

CREATE TABLE executive\_strengths (

id SERIAL PRIMARY KEY,

executive\_id INTEGER NOT NULL,

strength\_description TEXT NOT NULL,

display\_order INTEGER NOT NULL DEFAULT 0,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (executive\_id) REFERENCES executives(id) ON DELETE CASCADE

);

-- Create function to update updated\_at timestamp

CREATE OR REPLACE FUNCTION update\_updated\_at\_column()

RETURNS TRIGGER AS $$

BEGIN

NEW.updated\_at = CURRENT\_TIMESTAMP;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Create trigger for executives table

CREATE TRIGGER update\_executives\_updated\_at

BEFORE UPDATE ON executives

FOR EACH ROW

EXECUTE FUNCTION update\_updated\_at\_column();

Now, we can connect the springboot to the database so that the profiles will be shown thorugh a query, not through just static text.

Also, here I have a sample data insertion:  
  
INSERT INTO executives (gender, title, experience, sector\_focus, location)

VALUES (

'male',

'Fractional CFO | Strategic Financial Leadership',

'20+ years (CEO, board member, Big4 background)',

'Manufacturing, packaging, recycling, private equity',

'Romania (hybrid/remote)'

);

-- Insert highlights (PostgreSQL will automatically use the returned ID)

INSERT INTO executive\_highlights (executive\_id, position\_title, company\_description, details, display\_order)

VALUES

(1, 'CEO & President of the Board', 'Leading Circular Economy Group',

'Transformed loss-making firm into industry leader (€120M+ turnover)

Led €100M+ in investments (incl. EU & Norwegian grants)', 1),

(1, 'Board Member', 'Listed Private Equity Fund',

'Oversees €700M portfolio across banking, real estate, energy

Active in EU-level advocacy', 2),

(1, '10+ Years in Big4', 'Global Professional Services Firms',

'Specialized in financial services, IFRS, transformation projects

Led EU-funded training & advisory for regulators', 3),

(1, 'Proven M&A & Strategy Execution', NULL,

'Delivered growth through vertical integration and green finance

Scaled operations with focus on sustainability and innovation', 4);

INSERT INTO executive\_strengths (executive\_id, strength\_description, display\_order)

VALUES

(1, 'M&A and strategic growth execution', 1),

(1, 'Regulatory & financial services expertise', 2),

(1, 'Operational scaling & sustainability leadership', 3);

CI/CD Pipeline

I’ve created a pipeline that accomplishes the following:   
- Pulls the latest version of the codespace from the main branch on github  
- Build the app using Maven  
- Stops the old instance(if needed) and starts the new JAR

What I’ve used to accomplish this:  
-Github Actions  
-SSH key-based authentication  
-deployment script( deploy.sh)

My VM:  
-OS: Ubuntu  
-User: ubuntu  
-Project Path: /home/ubuntu/fraxen  
  
Steps:  
-Key generation: `ssh-keygen -t ed25519 -C "github-deploy"` this generated one public key and one private key.  
-I added the public key to vm, and the private one to github secrets.  
-I also tested the connection first using `ssh -i /path/to/id\_ed25519 ubuntu@<VM\_IP>` on my macbook terminal  
-Used the next deployment script:  
  
#!/bin/bash

set -e # exit on error

cd /home/ubuntu/fraxen || exit

echo ">>> Pulling latest changes..."

git pull origin main

echo ">>> Building project..."

mvn clean package -DskipTests

echo ">>> Stopping old app..."

pkill -f 'fraxen-0.0.1-SNAPSHOT.jar' || true

echo ">>> Starting new app..."

nohup java -jar target/fraxen-0.0.1-SNAPSHOT.jar > app.log 2>&1 &  
  
-I made the script executable  
-Created deploy.yml:  
name: Deploy to VM

on:

push:

branches:

- main

jobs:

deploy:

runs-on: ubuntu-latest

steps:

- name: Run deploy script on VM

uses: appleboy/ssh-action@v1.2.0

with:

host: ${{ secrets.VM\_HOST }}

username: ${{ secrets.VM\_USER }}

key: ${{ secrets.VM\_SSH\_KEY }}

script: |

cd /home/ubuntu/fraxen

./deploy.sh  
  
- I set up github secrets (won’t disclose).  
  
Some notes I found on the subject and I’ve considered as helpful:  
-SSH key-based authentication ensures secure, password-less deployment.   
-chmod +x deploy.sh is required so the script is executable.   
-nohup allows the Java app to run even if the SSH session terminates.   
-Permissions for the private key must be restricted (chmod 600), otherwise SSH ignores it.   
-For production, using a systemd service for the Java app is recommended for better management and auto-restarts.