DA Registry

Application Developer Guide

Digital Green

February, 2024

Acronyms

DA - Development Agent

EA – Extension Agent

MoA – Ministry of Agriculture

1. Introduction

1.1. Overview

This Developer guide provides general information for managing, maintaining, deploying and troubleshooting the DA Registry web application.

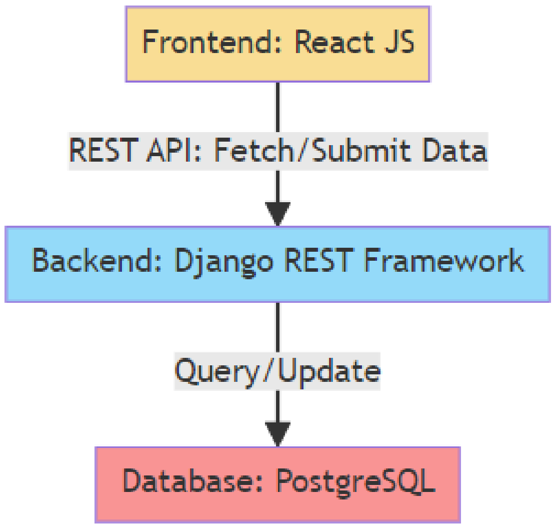
DA Registry is a digital data management system built for the purpose of managing and maintaining profiles of DAs within the extension department of MoA. This web-based application provides access to federal, regional, zonal, woreda and kebele level extension department users maintain DA profiles. In general, this platform will let the extension system digitize the process of DA profile management and creates a central pool of profiles that can be managed within each hierarchical level structures within MoA.

1.2. Audience

The primary audience for this guide is web application developers, maintainers, database administrators and IT professionals responsible for the upkeep of the DA Registry System.

2. System Overview

2.1. System Components and Technology Stack



2.2. Repositories

Frontend Repository	(https://github.com/digitalgreenorg/da_registry/)
Backend Repository	https://github.com/digitalgreenorg/da_registry/
Environments: Development: Staging Production Demo	 https://dev.digiext.org/ https://stage.digiext.org/ https://prod.digiext.org/ https://demo.digiext.org/

3. Environment Setup

3.1. Installation Instructions

3.1.1. Environment Prerequisites

- ✔ Ubuntu Server
- ✔ Docker and Docker Compose install
- ✔ Git Repository
- ✔ Jenkins
- ✔ Ansible
- ✔ web server

3.1.2. CI/CD Setup

- ✔ Code Repository: Developers commit their code changes to a Git repository. This repository serves as the source of truth for the project.
- ✔ Webhook/Trigger: Whenever a developer commits new code or pushes changes to the Git repository, a webhook or trigger is set up to notify Jenkins about the changes. This webhook can be configured to automatically initiate the CI/CD pipeline.
- ✔ Jenkins Server: Jenkins is used as the automation server to manage the CI/CD pipeline. It monitors the Git repository for changes.
- ✔ Source Code Pull: Jenkins detects the code changes and pulls the latest code from the Git repository to ensure it has the most up-to-date source code to work with.

- ✓ Docker Image Build: After pulling the latest code, Jenkins initiates a build process for creating a Docker image. This step involves compiling the code (if necessary), resolving dependencies, and creating a Docker image containing the application.
- ✓ Docker Image Push: Once the Docker image is built successfully, Jenkins pushes the newly created Docker image to a Docker Hub repository or another container registry. This step ensures that the latest version of the application is available for deployment.
- ✓ Ansible Playbook Execution: An Ansible playbook is triggered by Jenkins. This playbook is responsible for managing the deployment process on the target server(s). The playbook performs the following tasks: - Removes the previous version of the application or any existing containers. - Pulls the latest Docker image from Docker Hub or the container registry. - Starts the application using Docker Compose or any other suitable orchestration tool. This step can include setting environment variables, configuring network settings, and managing any other necessary setup.
- ✓ Notification: Jenkins can send notifications or reports to relevant team members or stakeholders, informing them about the status of the CI/CD pipeline, including whether the deployment was successful or encountered any issues.
- ✓ Pipeline Automation: The CI/CD pipeline is fully automated, reducing the need for manual intervention. This ensures that code changes are quickly and consistently deployed to the production environment.
- ✓ Location of environment variable files
  - env ui: /home/jenkins/env/env_dev_ui.sh
  - env be:/home/jenkins/env/env_dev_be.sh
  - env db:/home/jenkins/env/env_dev_db.sh
- ✓ Pipeline Configuration

```
pipeline {
  agent any
  stages {
    stage(&#39;Checkout the code&#39;) {
      steps {
        checkout([$class: &#39;GitSCM&#39;, branches: [[name: &#39;*/dev&#39;]], extensions: [],
userRemoteConfigs:
[[credentialsId: &#39;jenkins-git&#39;, url: &#39;https://github.com/digitalgreenorg/da_registry&#39;]]])
      }
    }

    stage(&#39;Build and Push Frontend Image&#39;) {
      steps {
        script {
          dir(&#39;src/frontend&#39;) {
            // Build and push frontend Docker image
```

```
                        sh &quot;docker build -t farmstack/ea-registry-dev-ui .&quot;
                        withCredentials([usernamePassword(credentialsId: &#39;02e85174-922d-4606-bb0b-
c7030dc54350&#39;, passwordVariable: &#39;DOCKER_PASSWORD&#39;, usernameVariable:
&#39;DOCKER_USER&#39;)]) {
                            sh &#39;docker login -u $DOCKER_USER -p $DOCKER_PASSWORD&#39;
                            sh &#39;docker push farmstack/ea-registry-dev-ui&#39;
                        }
                    }
                }
            }
        }

        stage(&#39;Build and Push Backend Image&#39;) {
            steps {
                script {
                    dir(&#39;src/backend&#39;) {
                        // Build and push backend Docker image
                        sh &quot;docker build -t farmstack/ea-registry-dev-be .&quot;
                        sh &#39;docker push farmstack/ea-registry-dev-be&#39;
                    }
                }
            }
        }

        stage(&#39;Triggering Ansible Playbook&#39;) {
            steps {
                script {
                    sh &#39;cp -r /home/jenkins/ansible/conf/* .&#39;
                    sh &#39;cp -r /home/jenkins/ansible/yaml-files/ea-registry/* /var/lib/jenkins/workspace/ea-
registry/.&#39;
                    sh &#39;ansible-playbook ea-registry-dev.yaml&#39;
                }
            }
        }
    }

    post {
        success {
            emailext (
                attachLog: true,
                body: &#39;Build is successfully deployed for ea-registry-dev....!&#39;,
                subject: &#39;$PROJECT_NAME - Build # $BUILD_NUMBER - $BUILD_STATUS!!&#39;,
                to: &#39;DA_UPD_dev@digitalgreen.org&#39;

            )
        }
        failure {
            emailext (
                attachLog: true,
                body: &#39;Build has failed for ea-registry-dev....!&#39;,
                subject: &#39;$PROJECT_NAME - Build # $BUILD_NUMBER - $BUILD_STATUS!!&#39;,
                to: &#39;DA_UPD_dev@digitalgreen.org&#39;
        )
      }
   }
}
```

4. Database Schema

5. API Documentation
    5.1. Endpoints
    5.2. Authentication
6. Deployment Process
    6.1. Build Instructions
    6.2. Deployment Steps
7. Versioning and Change Log
8. Contact Information
9. Appendix