Integración de Shiny con Aplicaciones Web Node.js para visualización y análisis de datos de empresas agropecuarias.

Anónimo

Palabras clave: Shiny - Docker - Node.js

FUCREA es una organización que nuclea a más de 500 productores agropecuarios de Uruguay. Estos productores se organizan en grupos de 10 o 20 empresas que tienen un asesor especializado. Los asesores relevan información anual de más de 100 indicadores productivos y económicos de cada empresa. El objetivo de este proyecto es facilitar el uso de estos datos para la toma de decisiones, generando análisis y visualizaciones que permitan a los usuarios sacar conclusiones valiosas y tomar decisiones a partir de la información disponible.

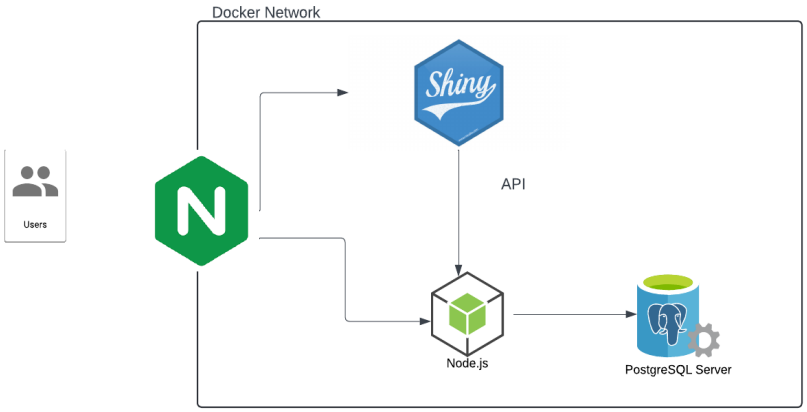
A nivel tecnológico, el principal desafío del proyecto fue diseñar un mecanismo de autorización que refleje la lógica de negocios de FUCREA y garantice que los usuarios solo acceden a la información para la que tienen permisos e integrar este sistema con un tablero de control con visualizaciones interactivas atractivas para la diversidad de usuarios que integran la organización. La solución final garantiza la seguridad y privacidad de los datos, implementando un modelo de autorización que refleja la lógica de negocio, y permite visualizarlos en una aplicación interactiva.

Las empresas CREA se organizan en grupos, que a su vez se agrupan en Sectoriales de empresas de rubros similares (ganaderas, lecheras, agrícolas, frutícolas, etc). Esta estructura jerárquica implica diferentes niveles de acceso para cada usuario del sistema. Hay un coordinador general que puede acceder a todos los datos, coordinadores sectoriales que tiene permisos solo a los de las empresas de su sector, asesores de grupo que solo pueden acceder a empresas de sus grupos y los productores solo pueden acceder a los datos de sus empresas. Además, hay asesores asignados a varios grupos y productores asignados a varias empresas. Estas relaciones son dinámicas y se administran por el equipo de FUCREA en una aplicación web que almacena los datos en una base de datos relacional PostgreSQL. En la primera parte de la presentación se presentará brevemente la funcionalidad de la aplicación web, especialmente los formularios para crear, actualizar y eliminar usuarios y empresas.

Por otro lado, los coordinadores, asesores y productores acceden a tableros de control desarrolladas con `shiny` y `shinydashboard`. Estas aplicaciones consumen información de la base de datos a través de una API. Las APIs definen protocolos para intercambiar datos en forma segura a través de internet (usualmente como objetos `json`).

Los tableros de control utilizan `shinymanager` y `httr` para autenticar a los usuarios con la API y enviar solicitudes para obtener los datos para los que tiene permisos de acceso cada usuario. Una vez descargados los objetos `json`, estos se convierten a `data.frames` para generar las visualizaciones con `ggplot`. En esta sección de la presentación se presentará detalladamente como autenticarse como usuario y hacer consultas a la API desde R usando las funciones `POST` y `GET` del paquete `httr`.

En producción, todos los componentes se encuentran detrás de un proxy Nginx, que centraliza los certificados SSL y direcciona el tráfico externo hacia las distintas aplicaciones. El siguiente diagrama muestra la relación entre los componentes del sistema y sus interacciones.



Configurar y mantener la infraestructura para que todos estos componentes funcionen correctamente en ambientes de desarrollo, testing y producción puede ser un costo considerable, especialmente para una organización que no tiene equipos TI dedicados. Docker permite automatizar este proceso y reducir considerablemente los requerimientos de administración de infraestructura. Todos los componentes del sistema corren en una instancia AWS con Docker como único requisito de software.

Una imagen Docker contiene todo el software necesario para correr una aplicación y la configuración necesaria para interactuar con otros sistemas. Shiny, PostgreSQL y Node.js tienen imagenes de Docker disponibles. Además, es posible personalizar las imágenes oficiales, agregando otras librerías y nuestras propias aplicaciones. Una vez creada la imágen, usamos Docker para crear un contenedor, un proceso que ejecuta el código de nuestra aplicación. En esta sección de la presentación se muestran los comandos `docker image build` y `docker container run`. Estos comandos permiten crear imagenes y contenedores Docker.

Docker Compose es una herramienta complementaria a Docker, que permite orquestar varios contenedores. Docker Compose centraliza en un único archivo la configuración de los componentes del sistema mediante un lenguaje declarativo. Esto facilita la administración de la infraestructura, permitiendo utilizar control de versiones y evitando el uso de scripts y variables de entorno para configurar unidades de almacenamiento, redes y puertos y toda la infraestructura necesaria para que los contenedores se ejecuten e interactúen correctamente. En la última sección, se presenta el archivo `docker-compose.yml`, donde se centraliza la configuración de la estructura del sistema.

Este proyecto integra las capacidades de análisis de R con una aplicación web Node.js para desarrollar una solución robusta y escalable, utilizando Docker y Docker Compose para simplificar la administración y garantizar la seguridad de los datos. A través de una solución que respeta la complejidad jerárquica y las necesidades de acceso diferenciado de los usuarios de FUCREA, se ha desarrollado una herramienta que facilita el uso de datos históricos valiosos para la toma de decisiones informada en las empresas CREA.