Dockerization of Legacy PERN Application

Dockerization of Postgres DB

Dockerfile

Created a Docker file for Postgres, which does the following:

1. Imports a Postgres image from the Docker Hub Container Image Library
2. Sets a Superuser password
3. Creates a Citius DB
4. Exposes port 5432

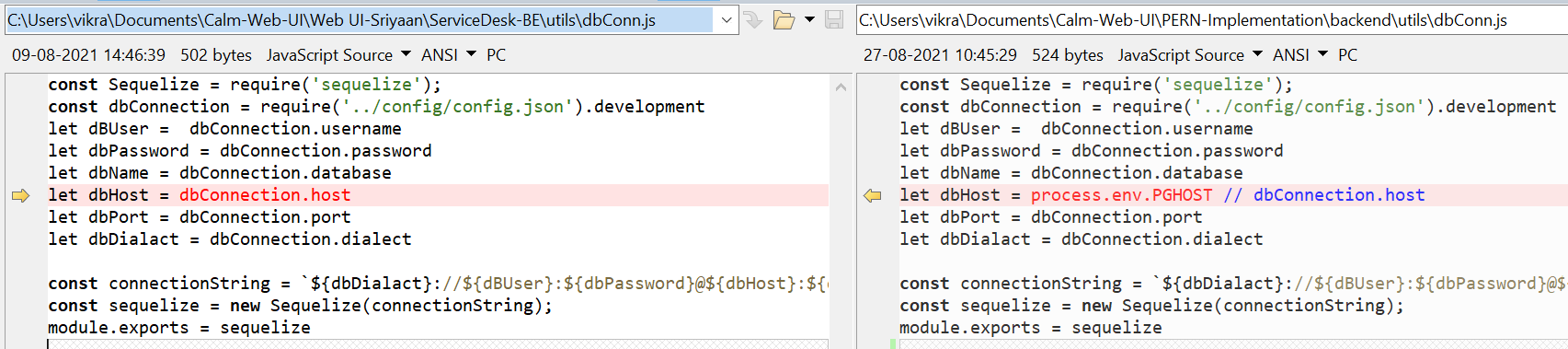
docker-compose.yml

1. It specifies a Volume to store the data
2. Creates a Postgres Container after building an image using the specific Dockerfile
3. It then exposes the 5432 internal-external port pair

Dockerization of Node Backend

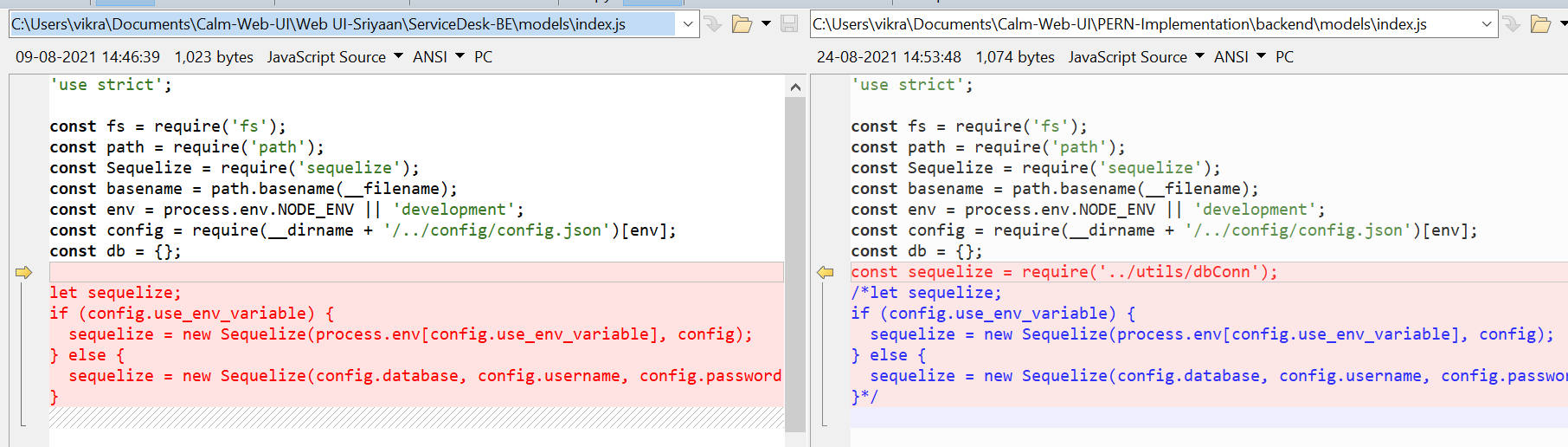
Dbconn.js

Used an Environment variable to fetch the Docker Service for Postgres - “**pg**”, instead of using localhost. The environment variable is set in docker-compose.yml



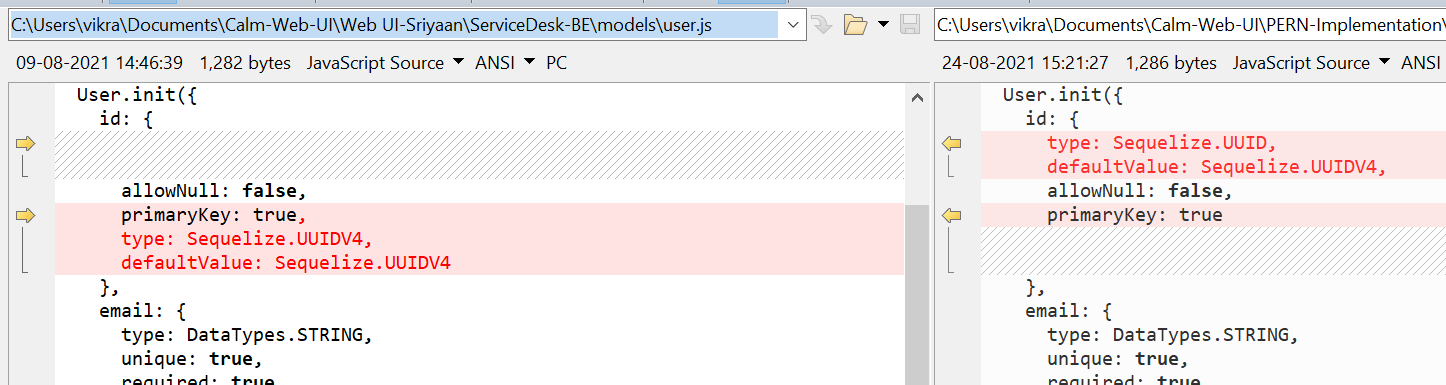
Models\index.js

Using the sequelize object from Dbconn, which uses Docker service as host instead of “localhost”

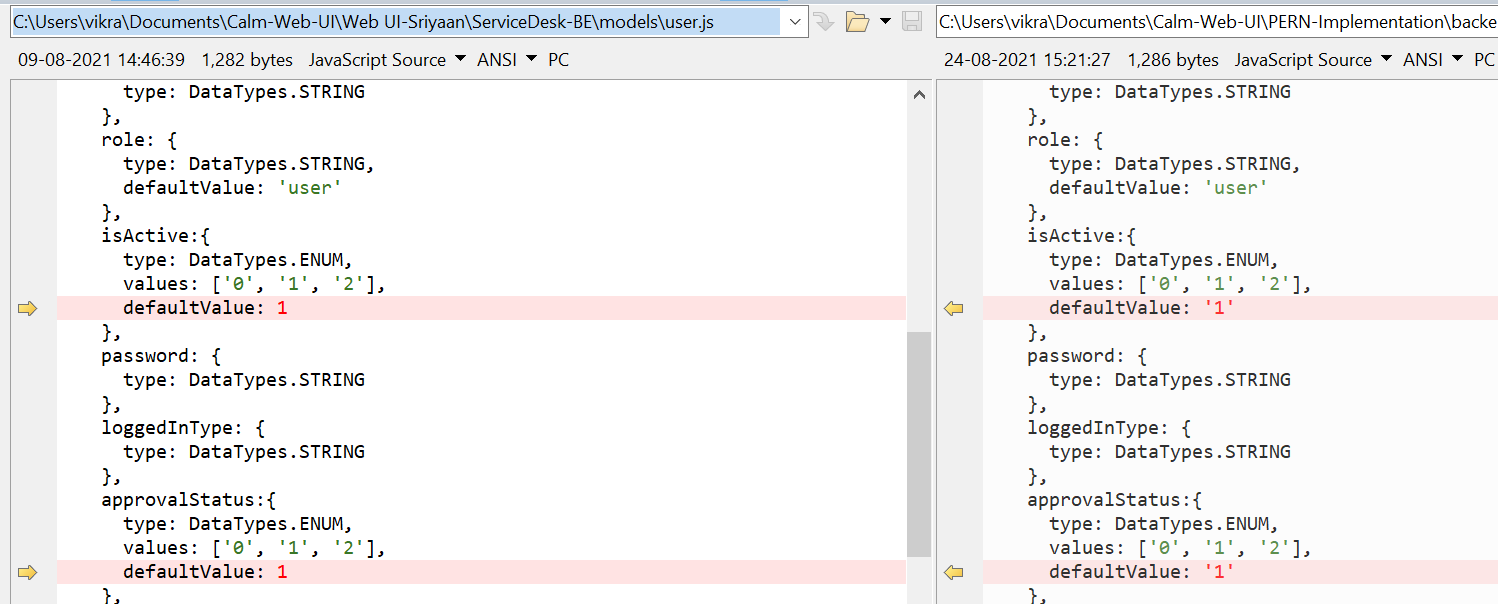


Models\user.js

Changed the type from **Sequelize.UUIDV4** to **Sequelize.UUID** for the User Id field



And changed the default value for the 2 enums from **int** to **string**



Dockerfile

Created a Docker file for the Node backend, which does the following:

1. Imports a Nodejs image from the Docker Hub Container Image Library
2. Copies over our application code
3. Runs “npm install” to fetch all the dependencies - express, sequelize, etc.
4. Exposes port 8088
5. Sets the main run Command to “node app.js”

docker-compose.yml

1. Creates a Backend Container after building an image using the specific Dockerfile
2. Sets the dependency and environment variable for the Postgres service
3. It then exposes the 8088 internal-external port pair