# Cấu trúc thư mục

## assets

Nơi đặt các file cv của các nhân viên khi cập nhật ở chức năng user-management

## config

Nơi đặt file cấu hình connect cho database.

File database.js đặt cấu hình connect bằng knex

File database.pg-promise.js đặt cấu hình connect bằng thư viện pg-promise

## constants

Có file chứa các cấu hình các biến toàn project

## controller

Đặt các file controller – các file để tiếp nhận request và tương tác với các repository để xử lý công việc.

## database

Đặt cấu hình các database.

-Bên trong là đặt các file sql để tương tác với query tool trong trường hợp migration của knex không hoạt động.

-migrations: Chứa các file migration sẽ được knex module sử dụng tạo các table tương ứng. Ở thời điểm hiện tại của file, chỉ hỗ trợ migrations từ nghiệp vụ asset trở đi. Các table về nghiệp vụ quản lý người dùng trước file 20200626-xxx là phải copy – paste vào query tool trước xong mới chạy knex-it được.

+Để tạo các file mới thì có thể dụng lệnh **yarn knex migrate:make <file-name>** với <file-name> là tên của file muốn tạo. Không nên tạo bằng tay vì nó đã tự động thêm timestamp cho file rồi.

-seeds: Dùng để init một vài table khi đã chạy knex-it xong – khuyến nghị không sử dụng tiếp ở đây. Tất cả việc init vào table nên được xử lý trong migrations. **Triết lý:** Chỉ tin tưởng và sử dụng migrations file.

## entity

Đặt các file xử lý sau query db.

## enviroments

Đặt các file cấu hình môi trường. Copy – paste file .env.sample và đổi tên thành .env và điền đầy đủ thông tin tương ứng.

## middleware

Trong đây có hai phần quan trọng đấy là **validator** và file **error.js**

**validator** là thư mục để validate data trước khi đưa vào xử lý nếu cần thiết

**error.js** là file chứa các error handling cho toàn bộ project

## repository

Đặt các file query vào cơ sở dữ liệu

-Thư mục **jira-repository** đặt các file query đến hệ thống api của jira.

-Thư mục **postgres-repository** đặt các file tương tác với csdl postgres sql.

- Ghi chú: Phiên bản của pg trên server qcd là **9.4**

## routes

Đặt các file hứng các request từ client

## service

Đặt các dịch vụ phục vụ đằng sau controller. Phục vụ cho các controller xử lý các tác vụ.

## utils

Nếu cần xử lý gì đó phức tạp. Hãy đặt nó ở đây.

-Thư mục ngoài lề: **APIException** không chỉ là utilities. Nó chứa các cấu trúc class xử lý lỗi của ứng dụng sẽ nói về sau.

# Cài đặt server lần đầu

-Vì đặc thù và đã nói trong cấu trúc thư mục của database. Nếu muốn chạy ứng dụng lần đầu thì cần làm như sau:

+B1: Xác định trong thư mục database/migrations có file **20200728134902\_init-user-tables.js, 20200728134903\_init-table-role.js hay chưa?**

+B2: Nếu đã có hai file kể trên, chỉ cần chạy **yarn knex-it và yarn-knex-seed**  là database đã sẵn sàng phục vụ. Các bước setup đã xong, chuyển đến **B5**

+B3 Nếu chưa có hai file kể trên tức là branch feature/SYS-1 chưa được merge vào develop có thể hỏi leader về việc merge branch này.

+B4 Nếu chưa thể có ngay và cần database sử dụng: Copy và paste các file sql trước file **20200626-init-asset-manager-table.sql** (không kể file này) vào query tool. Tức là phải tạo các bảng về việc quản lý user bằng tay còn từ chức năng quản lý tài sản thì mới có migrations.

+B5: Thử vào database client để xem liệu có các bảng hay không.

# Cơ chế error handling

Cơ chế được mô tả bằng việc sử dụng các file trong thư mục **utils/APIException**. Trong đó:

-File **ExtendableError.js** là class cơ bản để tạo các error cho api.

-Nếu muốn custom các error cho user thì sử dụng file **APIError.js**

-Nếu muốn bắt các query từ knex => sử dụng **APIErrorWithKnex.js** để bắt.

-Các ví dụ sử dụng: Nếu dùng vscode thì search toàn project các file tương ứng để xem ví dụ.

-Các step của error handling:

+Bắt lỗi 404: Nếu mà không có các route tương ứng => Trả về lỗi 404

+Bắt lỗi authen/author của thư viện express-jwt ném ra.

+Bắt lỗi query bằng knex với class **APIErrorWithKnex**

+Bắt lỗi developer custom bằng class **APIError**

-Nếu có các định nghĩa lỗi khác có thể cân nhắc sử dụng trước lớp bắt bằng knex sẽ tiện hơn.

# Cấu trúc response trả về client

Các cấu trúc cũ đang sử dụng thư viện response-format cho data trả về client. Nên là mã lỗi luôn là 200 cho tất cả trường hợp. Vì theo hướng này, nuxt sẽ không tự động xử lý cho các lỗi ví dụ: 5xx phát sinh thì auto nuxt sẽ handler.

-Cấu trúc lỗi bình thường gặp có thể thấy tại file **utils/APIException/handleErrorKnex.js**

-Mô tả cụ thể một lỗi: Formatter.badRequest("error\_data\_invalid", "record\_contrain\_null\_value")

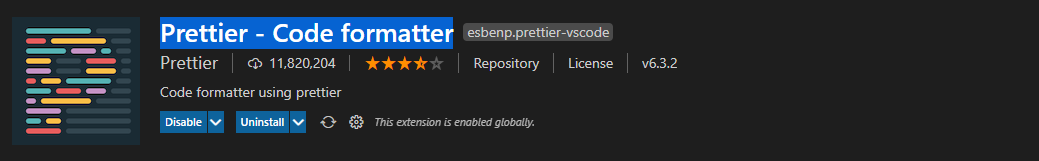
+Chú thích: badRequest là cách để tạo ra một object với mã lỗi 400.

+ error\_data\_invalid: Key hoặc đoạn văn bản sẽ hiển thị cho client bằng notification của ant-design. Cẩn thận khi sử dụng vì nếu đã xác định key ở đây thì phải define vào sheet trên drive để client có thể cập nhật tương ứng.

+ record\_contrain\_null\_value: Cũng tương tự, là key cho phần description của notification bên client đã được define sẵn.

+ đường dẫn đến file sheet: <https://docs.google.com/spreadsheets/d/1lLpjuQZLXLO1ZkKxpw0FRyw-1cI3PLGFNnwAHE8NEVY/edit#gid=0> hoặc có thể tìm trong repo của client

# Code convention

Mặc định project đã được cấu hình cho vscode và extension Prettier

Khuyến khích sử dụng extension để format code cho đồng bộ toàn project

# Commit convention

Các commit được có cấu trúc: <Task> <index> <message>

Trong đó **Task** là các nhiệm vụ được giao hoặc tạo trên trello, jira. **index** là số thứ tự kể từ khi bắt đầu làm task đó. **message** là một nội dung cần ghi lại.