Front end

1 mensaje

Eric Hidalgo <ehidalgo@sursoftware.com.ar> mar, 18 de jul de 2023 a la hora 01:05
Para: Juan Manuel Cortez <jcortez@sursoftware.com.ar>

Back-end

Se va utilizar NestJS 10. En el ejemplo de carpetas puse como ORM a TypeORM, pero queda en gustos cuál utilizar. La estructura de carpetas va a ser basada en una arquitectura Hexagonal (más info):

- /src/

- app.module.ts
- main.ts
- /domain
  - /user
    - /case
      - get-user-by-id-imple.ts
      - get-users-imple.ts
    - /model
      - user.ts
      - role.ts
    - /port
      - get-user-by-id.ts
      - get-users.ts
      - user-repository.ts
  - /domain2
  - /domain3
  - ...etc
- /infraestructure
  - /user
    - /rest
      - /input
        - user-filters.ts
      - /payload
        - user-payload.ts
      - /mapper
        - user-rest-mapper.ts
      - /controller
        - user-controller.ts
    - /typeorm
      - /mapper
        - user-typeorm-mapper.ts
        - role-typeorm-mapper.ts
      - /repository
        - user-typeorm-repository.ts
      - /model
        - user-typeorm.ts
        - role-typeorm.ts
  - /domain2
  - /domain3
  - ...etc

/domain

En esta carpeta debería ir el modelo de dominio, la lógica de negocios. Dentro de esta carpeta no debería haber librerías, nada relacionado a ORM's, API's, Queues, etc. Solo se pueden usar librerías de utilidad, como DayJs, BigJs o similares. Los modelos de dominios que existen dentro de esta carpeta no siempre son anémicos. Toda lógica que se repita en varios casos de uso debería ir en un modelo de dominio.

/infraestructure
En esta carpeta debería ir todos los adaptadores, como el ORM, la API, etc. La capa de infraestructura se comunica con el dominio mediante puertos que abstraen la implementación. Cada adaptador es responsable de modelar sus datos al modelo de dominio correspondiente, de ser necesario. Ejemplo: los repositorios, que son los encargados de comunicarse con la base de datos, son los encargados de adaptar sus datos al modelo de dominio. Lo mismo con los controllers de la API Rest.

/domain/{domain-name}/case
Aquí van los casos de uso, que contienen las reglas de negocio específicas de la aplicación. Estos casos de uso orquestan el flujo de datos hacia y desde los modelos, y dirigen a esos modelos para que utilicen sus reglas de negocio.

/domain/{domain-name}/model
Aquí van los modelos de dominio. Recordar que no debe tener decoradores relacionados a la persistencia o a la API. No saben nada de implementaciones.

/domain/{domain-name}/port
Aquí van los puertos de entrada y salida. Los puertos de entrada son interfaces que van a ser implementados por los casos de uso. En cambio, los puertos de salida son interfaces que van a ser implementados por los adaptadores, por ejemplo, los repositorios. No todos los adaptadores implementan un puerto, por ejemplo, los controllers no implementan ningún puerto. Los adaptadores de entrada no implementan puertos, se comunican con el dominio mediante un puerto.

/infraestructure/{domain-name}/rest/input
Aquí van los objetos que pueden ser input en un endpoint, por ejemplo, para crear un usuario necesitamos un create-user-input.ts.

/infraestructure/{domain-name}/rest/payload
Aquí van los objetos que pueden ser respuesta en un endpoint, por ejemplo, si tenemos un endpoint que retorna un usuario, podemos tener un user-payload.ts. Los modelos de dominios no deberían exteriorizarse al mundo exterior.

/infraestructure/{domain-name}/rest/mapper
Aquí van los mappers que veamos necesarios. Recordar que los modelos de dominios no deberían exteriorizarse al mundo exterior, por eso es posible que necesitemos mappers para transformar un modelo de dominio a un payload, o un input a un modelo de dominio.

/infraestructure/{domain-name}/rest/controller
Aquí van los controllers de nuestra API Rest. Cada controller debe encargarse de mapear los datos desde y hacia el dominio correspondiente.

/infraestructure/{domain-name}/typeorm/model
Aquí van los objetos que modelan la base de datos. Son nuestras entidades, que contienen los decoradores necesarios para saber cómo persistir y recuperar los datos.

/infraestructure/{domain-name}/typeorm/repository
Aquí van nuestros repositorios, encargados de realizar las queries y CRUD's correspondientes contra nuestra base de datos.

/infraestructure/{domain-name}/typeorm/mapper
Aquí van los mappers que veamos necesarios. Recordar que no siempre el modelo de dominio va a coincidir con nuestro modelo de ORM, por eso es que posiblemente necesitemos un mapper que haga ese trabajo.
[Texto citado oculto]