Transportation-accounting-api

Technical document - 2024-09-09

Introduccion

El objetivo de este documento es especificar cuales son las funcionalidades de nuestra Api. Esta se llama Transportation-Accounting-Api, nos permite crear presupuesto para una empresa específica, teniendo en cuenta los productos brindados, los trabajos realizados y aquellos por hacer. Dentro del presupuesto vamos a encontrar opciones determinadas como un trabajo por horas o por viajes. Una de las opciones también vamos a encontrar un presupuesto para saber las ganancias obtenidas dentro de cada trabajo asignado.

Requerimientos del trabajo

Para el cierre de este módulo, el desafío integrador consiste en desarrollar una API REST siguiendo los lineamientos que vimos a lo largo de las clases. La consigna es desarrollar un backend que permita automatizar las tareas de algún trabajo que ustedes elijan, como por ejemplo un sistema de gestión de stock de un local que comercia mercadería, o para administrar los pedidos de algún restaurant.
Tengan en cuenta que este proyecto va a conformar su primer proyecto bastante completo y profesional, así que les recomiendo seguir los siguientes lineamientos, los cuales fui recopilando de distintos desafíos técnicos reales que las empresas usan para evaluar candidatos.

Ítems a tener en cuenta

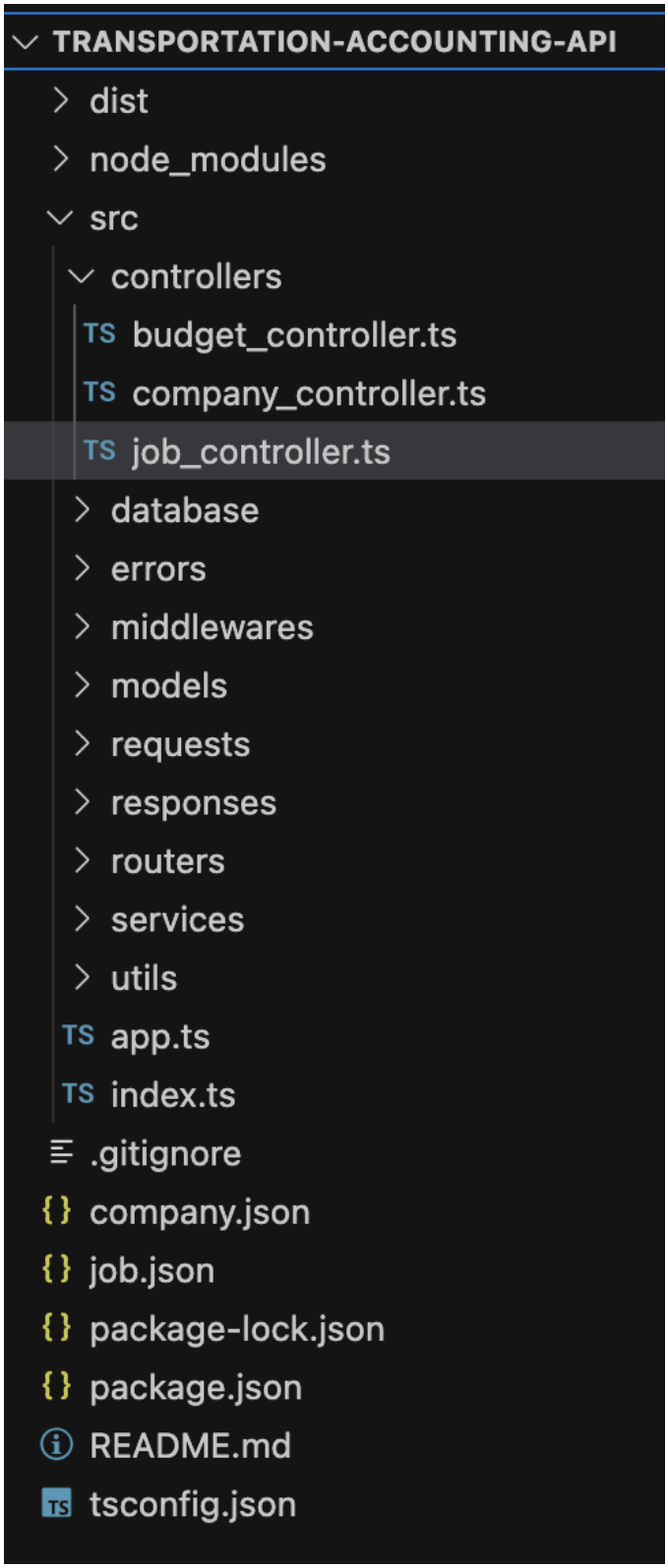
- Endpoint de ping que devuelve estado del servidor y versión.
- Hasheo de password al registrar usuario.
- Utilizar middleware para validar el token.
- Enviar el token desde el cliente en el header que corresponda.
- Crear y configurar archivo de configuración y carga de variables de entorno.
- NO COMMITEAR NI PUSHEAR LAS VARIABLES DE ENTORNO: Malísima práctica de seguridad.
- Documentar la API de forma clara.
    - La documentación no tiene que repetir cosas que están en el enunciado. Tiene que ser clara y concisa, fácil y rápida de entender.
    - Debe tener suficientes ejemplos de uso para todos los casos. Cosa de que el usuario pueda probar directamente esos comandos y ver los

resultados esperados.

- Tiene que haber una clara distinción de capas y responsabilidades. Cada capa debe tener la lógica correspondiente.
- Los códigos de estado deben estar correctamente utilizados, según corresponda a cada situación.
- Las funciones/métodos debe retornar siempre el mismo tipo de dato, no debería retornar un número o un objeto dependiendo del caso. Para casos de error, lo mejor es arrojar una excepción/error. Esto es para mantener la coherencia y la inteligibilidad del código.
- Los sistemas de filtrado para las búsquedas deberían:
    - Ser flexibles y permitir varios parámetros en simultáneo.
    - Las consultas hechas tienen que ser case insensitive. El usuario debería tener la menor cantidad de consideraciones posibles a la hora de realizarlas.
- Tratar de agrupar lógica para no repetir código (DRY).
- Tratar de tener pocas fuentes de la verdad, cosa de cambiar el código en un sólo punto.
- Detenerse siempre a pensar bien el nombre de las variables/funciones.
- Cada función debe tener una tarea en específico y cumplirla, siendo coherente con su nombre.
- KISS: Keep it simple, stupid.
    - Evitar la complejidad innecesaria.
    - Mantener el código minimalista, con lo necesario.
    - Separar bien el código por intereses. Desarrollar de forma modular, bien discriminado.
- Tratar de escribir el código bien declarativo, para más claridad.
    - Tratar de leer la documentación para entender bien a fondo lo que hago.
    - Investigar en internet para mejorar la calidad de su trabajo.
    - Uso estándares de nomenclatura.

Estructura del trabajo

De acuerdo a lo que se requiere para el trabajo,se armó este tipo de estructura:



El proyecto fue modularizado en las siguientes carpetas:

- Controllers: Dentro de esta carpeta vamos a encontrar los archivos relacionados a las diversas funciones, en este caso son “budget_controllers.ts”, “company_controllers.ts” y “job_controllers.ts”.
- database: En esta carpeta se encuentra el archivo “database_file.ts” lógica de guardado, escritura y lectura en Json.
- Models: Tiene los archivos “company_model.ts”, “job_model.ts” y “job.ts” con las interfaces y clases asociadas a los modelos. Cada clase posee la conexión con la database.
- Error: Esta carpeta tiene un archivo llamado “general_error.ts” la cual contiene una clase con errores generales y un constructor.
- Requests: Esta carpeta contiene los archivos “budget_request.ts”, “company_request.ts” y “job_request.ts” en estas vamos a encontrar las validaciones creadas con la librería zod.
- Responses: Dentro de esta vamos a encontrar los archivos “budget_responde.ts” “company_response.ts” y “job_response.ts”, allí vamos a encontrar las interfaces con los datos que se requieren para dar una respuesta.
- Middlewares: Esta carpeta contiene los archivos “authoryzation_handler.ts” y “error_handler.ts” con las funciones del error handler.
- routers: En esta carpeta se encuentran los siguientes archivos: “budget_router.ts” “company_router.ts” y “job_router.ts” “index.ts” Acá se encuentran las rutas a las distintas funcionalidades y dentro de index.ts las subrutas.
- Utils: Dentro de esta carpeta vamos a encontrar el archivo “entity.ts”
- Index: Allí se encuentra el puerto en el que corre y escucha la app.
- En el archivo app.ts vamos a encontrar la estructura de los principales routers.
- El proyecto corre en el puerto 8080.

Lógica del trabajo

Primero se deben de cargar los datos de base, como los recursos, y también los costos fijos.Como serían la cantidad de camiones,que tipo de camiones,los costos del chofer por día,al igual como con las máquinas,que tipo de máquina y también el costo del chofer por día.

Tenemos una lógica extra que es para que se cree una empresa pidiendo contraseña y razón social. Esto te devuelve un token de acceso, si el token ya existe retorna los datos de la empresa y no la vuelve a crear. (Esto se guarda en la base de datos)

Dentro de la carpeta Services vamos a encontrar la lógica principal de negocio como

por ejemplo: calcular el porcentaje de las ganancias que se quiere obtener de un trabajo, teniendo en cuenta los recursos(máquinas,camiones,choferes,etc), los costos fijos y costos variables, terminando así con un precio total entre el porcentaje de ganancia y los costos extras (fijos y variables).

Es para crear una compañía:

```
curl --location 'http://localhost:8080/api/company' \
--header 'Content-Type: application/json' \
--data '{
    "name": "diaz tp final 2",
    "password": "diaz 1234"
}'
```

Este es para obtener el token:

```
curl --location 'http://localhost:8080/api/company/token?code=DIAZTPFINAL2' \
--header 'Password: diaz 1234'
```

Este es para crear el presupuesto:

```
curl --location 'http://localhost:8080/api/budget' \
--header 'Content-Type: application/json' \
--data '{
    "fixedCosts": [
        {
            "amount": 10,
            "description": "costo fijo",
            "resourceType": "OTHER"
        }
    ],
    "variableCosts": [
        {
            "costs": [
```

```
            {
                "amount": 90,
                "description": "un costo"
            }
        ],
        "resourceType": "TRUCK",
        "description": "costo de los camiones"
    }
  ],
  "hoursPerDay": 8,
  "totalDays": 1,
  "unitType": "HOUR",
  "profitPercentage": 30
}'
```

Este es para crear un job:

```
curl --location 'http://localhost:8080/api/job' \
--header 'Content-Type: application/json' \
--data '{
  "from": "2024-09-08",
  "to": "2024-10-08",
  "budget": {
    "fixedCosts": [
      {
        "amount": 10,
        "description": "costo fijo",
        "resourceType": "OTHER"
      }
    ],
    "variableCosts": [
      {
        "costs": [
          {
            "amount": 90,
            "description": "un costo"
          }
        ],
        "resourceType": "TRUCK",
        "description": "costo de los camiones"
      }
    ],
    "hoursPerDay": 8,
    "totalDays": 1,
    "unitType": "HOUR",
```

```
        "profitPercentage": 30
    }
}'
```

Get all jobs:

```
curl --location 'http://localhost:8080/api/job/all' \
--header 'X-Authorization: d3e8ff7d-cbdf-47b7-9ff3-7c0fde6c51ab' \
--header 'X-Company-Code: DIAZTPFINAL1'
```

Get by id de los jobs:

```
curl --location 'http://localhost:8080/api/job/b1176326-2812-41cf-b43f-412df507a228' \
--header 'X-Authorization: d3e8ff7d-cbdf-47b7-9ff3-7c0fde6c51ab' \
--header 'X-Company-Code: DIAZTPFINAL1'
```