ChatGPT

Develop Database Schemas, Controllers, and Routes for nap-serv

You're extending the **nap-serv** backend, a multi-tenant PERN-stack project for cost and profitability. Below are the conventions, patterns, and tasks you must follow.

Project conventions

- **Environment**: Node.js (ES modules), Express, PostgreSQL, per-tenant schemas (schema-qualified queries).
- **Module folder scaffold**: Each module follows this structure:
- `apiRoutes/v1`: Express routes using the `createRouter` utility.
- `controllers`: PascalCase filenames, extending `BaseController`.
- `middlewares`: Middleware functions specific to the module.
- `models`: PascalCase filenames, extending `TableModel` from `pg-schemata`.
- `schema`: Table definitions using the `TableSchema` pattern.
- `utils`: Utility functions specific to the module.
- **Schemas**: Define database tables in the `schema` folder; each table includes a `tenant_code` column. This column is used only for cross-tenant administration queries and is ignored in in-tenant queries, indexes, foreign keys and unique constraints. Tenant schemas remain independent.
- **SQL**: Always qualify queries with the tenant's schema (e.g., `SELECT * FROM ${tenantSchema}.TableName`). Use `tenant_code` for super-admin queries across tenants, but do not rely on it within tenant-specific models and controllers.
- **Imports/Exports**: Use ES6 syntax. Repository objects use camelCase keys.
- **Existing examples**: Refer to `CatalogSkus.js` and `CatalogSkusController.js` for guidance.

Required tasks

1. **Write a schema and create a table using** `pg-schemata`
2. In the `schema` folder, create a file exporting a class extending `TableSchema`.
3. Define columns, types, and constraints; include a `tenant_code` field but ensure it's not used in indexes, foreign keys, or unique constraints within the tenant schema.

4. Provide a migration or initialization function that creates the table within the correct tenant's schema.

5. **Implement a controller pattern**

6. In `controllers`, create a PascalCase file extending `BaseController`.

7. Pass the model name to `super()` in the constructor. Implement CRUD methods and any custom methods (e.g., file import) following the `CatalogSkusController` pattern:

```
class ExampleController extends BaseController {
  constructor() {
    super('exampleTable');
  }
  async getAll(req, res) {
    try {
      const data = await new ExampleModel().getAll(req?.tenantId);
      res.json(data);
    } catch (err) {
      this.error(res, err);
    }
  }
  // other methods as needed…
}
export default new ExampleController();
```

8. **Implement routing pattern via** `createRouter`

9. Use the `apiRoutes/v1` folder to define Express routers.
10. For most controllers, export a simple router using `createRouter(controllerInstance)`:

```
import exampleController from '../../controllers/ExampleController.js';
import createRouter from '../../../src/utils/createRouter.js';

export default createRouter(exampleController);
```

11. When additional routes or middleware are needed, pass a customization function and middleware options:

```
import tenantsController from '../../controllers/TenantsController.js';
import createRouter from '../../../../src/utils/createRouter.js';
import { requireNapsoftTenant } from '../../../../middlewares/access/
requireNapsoftTenant.js';

export default createRouter(
  tenantsController,
  router => {
    router.route('/:id/modules').get((req, res) =>
      tenantsController.getAllAllowedModules(req, res)
    );
  },
  {
    postMiddlewares: [requireNapsoftTenant],
    getMiddlewares: [requireNapsoftTenant],
```

```
    putMiddlewares: [requireNapsoftTenant],
    deleteMiddlewares: [requireNapsoftTenant],
    patchMiddlewares: [requireNapsoftTenant],
  }
);
```

12. **Review and feedback**

13. Before adding new files, inspect existing models, controllers, and routes to match the project's style and patterns.
14. Ensure that all queries are schema-qualified and that `tenant_code` is properly handled for cross-tenant scenarios.
15. Note and correct any deviations from project standards.

Note: This project forbids altering the independent nature of tenant schemas. Your response should be direct and to the point, without unnecessary praise.
