TECHNICAL SPECIFICATION DOCUMENT

Tax Types Management

Tax Types Management

VERSION	STATUS	CLASSIFICATION
1.0.0	Approved	Internal

Access Restricted to: Administrator Role

01 Overview

This document provides the complete technical and functional specification for the Tax Types Management module. It defines the data model, business rules, API contract, and operational flows required for implementation, review, and integration by engineering, QA, and product teams.

The module exposes a CRUD interface exclusively to users with the Administrator role, enabling the management of tax type records used across the broader financial system. All delete operations are non-destructive: records are logically flagged and excluded from standard queries without physical removal.

SCOPE

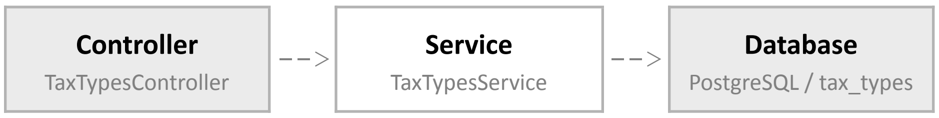
This specification covers the Tax Types sub-module only. Dependencies on other modules (e.g., invoice processing, tax calculation engine) are out of scope and addressed in separate documents.

AUDIENCE

- Backend Engineers — implementation and code review.
- QA Engineers — test case design and acceptance validation.
- Tech Leads / Architects — architectural review and sign-off.
- Product Managers — functional alignment and scope confirmation.

02 Architecture

The module follows a layered MVC architecture with a dedicated service layer. Each layer has a single, clearly bounded responsibility. No layer may communicate with a layer more than one level away — the Controller may not access the database directly.



LAYER	CLASS / COMPONENT	RESPONSIBILITY
Controller	`TaxTypesController`	Receives and validates HTTP requests. Delegates all business logic to the service layer. Maps service output to HTTP responses.
Service	`TaxTypesService`	Enforces business rules (uniqueness, soft-delete). Coordinates reads and writes with the repository.
Repository	`TaxTypesRepository`	Abstracts database access. Applies the isDeleted = false filter on all read operations.
Database	`PostgreSQL`	Persistent storage layer. Table: tax_types. No stored procedures or triggers; all logic resides in the application layer.

03 Functional Requirements

The following table enumerates all functional requirements for this module. Each requirement includes an unambiguous acceptance criterion that must be satisfied for the requirement to be considered complete.

ID	NAME	METHOD	ACCEPTANCE CRITERIA
RF-001	**Create Tax Type**	`POST`	System must allow creation of a tax type with a unique code and a required name. Response includes generated id, createdAt, and updatedAt.
RF-002	**List Tax Types**	`GET`	System must return all records where isDeleted = false. Deleted records must not appear in any list response.
RF-003	**Get Tax Type by ID**	`GET`	System must return the tax type matching the requested id. If not found or deleted, the response must be 404 Not Found.
RF-004	**Update Tax Type**	`PUT`	System must allow modification of the code and/or name fields. The uniqueness constraint on code must be re-evaluated on update.
RF-005	**Delete Tax Type**	`DELETE`	System must perform a soft delete by setting isDeleted = true. No record may be physically removed from the database.

04 Business Rules

Business rules define constraints and behaviors that must be enforced at the service layer, independent of the transport protocol. They are not optional and must be validated in both unit and integration test suites.

ID	NAME	RULE DEFINITION
RN-001	**Unique Code**	The code field must be unique across all records where isDeleted = false. An attempt to insert or update a record with a duplicate code must return a 409 Conflict error.
RN-002	**Soft Delete**	Records must never be physically removed from the database. The delete operation must update isDeleted to true and refresh the updatedAt timestamp.
RN-003	**Automatic Filters**	All SELECT queries must include a WHERE isDeleted = false clause. Records with isDeleted = true must be invisible to all standard API operations.
RN-004	**Automatic Dates**	The createdAt field is set once on INSERT and must not be modifiable afterward. The updatedAt field must be refreshed automatically on every UPDATE operation.

05 Data Model

The module operates on a single database table, tax_types. The table is append-only from a delete perspective — records are never physically removed. All schema migrations must be backward-compatible.

COLUMN	DATA TYPE	CONSTRAINT	NOTES
id	INTEGER	**PK**	Auto-increment primary key
code	VARCHAR(50)	**UNIQUE NN**	Business identifier for the tax type
name	VARCHAR(100)	**NOT NULL**	Human-readable display name
createdAt	DATETIME	**DEFAULT**	Auto-set on INSERT
updatedAt	DATETIME	**DEFAULT**	Auto-updated on UPDATE
isDeleted	BOOLEAN	**DEFAULT**	Soft-delete flag; default false

Indexes

INDEX NAME	COLUMN(S)	PURPOSE
PK_tax_types	id	Primary key. Clustered index; guarantees row-level uniqueness.
UQ_tax_types_code	code	Unique constraint. Prevents duplicate business identifiers.
IDX_tax_types_isDeleted	isDeleted	Partial index on active records. Improves query performance for filtered reads.

06 Input Validation

Input validation is enforced at the Controller layer using a DTO schema. Requests that fail validation must be rejected with a 400 Bad Request response before reaching the service layer. The following table specifies the validation rules for each writable field.

FIELD	REQUIRED	TYPE	MAX LEN.	CONSTRAINT / NOTES
code	Yes	String	50	Must be unique across all non-deleted records
name	Yes	String	100	Required; no additional uniqueness constraint
isDeleted	Auto	Boolean	—	Managed by the system; defaults to false
createdAt	Auto	DateTime	—	Set automatically on INSERT; not modifiable
updatedAt	Auto	DateTime	—	Updated automatically on each UPDATE operation

07 API Contract

All endpoints operate under the base path /api/v1. Requests must include a valid authentication token with the Administrator role. The API accepts and returns JSON payloads with Content-Type: application/json.

METHOD	ENDPOINT	RESPONSE	DESCRIPTION
GET	`/tax-types`	`200 OK`	Returns list of all active tax types (isDeleted = false)
GET	`/tax-types/:id`	`200 / 404`	Returns a single record; 404 if not found or deleted
POST	`/tax-types`	`201 Created`	Creates new record; validates uniqueness of code field
PUT	`/tax-types/:id`	`200 OK`	Updates code and/or name; validates uniqueness constraint
DELETE	`/tax-types/:id`	`200 OK`	Soft delete: sets isDeleted = true; no physical removal

Operation Flows

The tables below describe the internal call sequence for the two most operationally significant flows: record creation and soft deletion.

CREATE — POST /TAX-TYPES

ACTOR	REQUEST	CODE	RESPONSE / ACTION
Administrator	Sends POST /api/v1/tax-types with { code, name }	**POST**	HTTP request received by the Controller
Controller	Validates DTO structure and field constraints	**400**	Returns 400 Bad Request if validation fails; proceeds otherwise
Service	Checks code uniqueness: SELECT WHERE code = ? AND isDeleted = false	**409**	Returns 409 Conflict if duplicate; proceeds otherwise
Repository	Executes INSERT INTO tax_types	**INSERT**	Persists new record; database sets createdAt and updatedAt
Controller	Maps entity to ResponseDto	**201**	Returns 201 Created with the serialized record

SOFT DELETE — DELETE /TAX-TYPES/:ID

ACTOR	REQUEST	CODE	RESPONSE / ACTION
Administrator	Sends DELETE /api/v1/tax-types/:id	`DELETE`	HTTP request received by the Controller
Service	Fetches record by id WHERE isDeleted = false	`404`	Returns 404 Not Found if record does not exist or is already deleted
Service	Sets isDeleted = true and refreshes updatedAt	`UPDATE`	Business rule RN-002 applied; no physical removal
Repository	Executes UPDATE tax_types SET isDeleted = true, updatedAt = now()	`OK`	Persists the change
Controller	Returns empty body	`200`	Returns 200 OK with no response body

08 Security & Access Control

Access to this module is restricted to the Administrator role. All endpoints must be protected by the authentication and authorization middleware. Unauthenticated requests must return 401 Unauthorized. Authenticated requests from non-Administrator users must return 403 Forbidden.

ROLE	ACCESS	NOTES
Administrator	Full access	May perform all CRUD operations on all tax type records.
User (standard)	Denied — 403	No read or write access to this module.
Guest / Unauthenticated	Denied — 401	Request is rejected at the authentication middleware before reaching the controller.

09 Revision History

VERSION	DATE	AUTHOR	CHANGES
1.0.0	2026-02-21	Engineering Team	Initial release.