Milestone Four – Database Enhancement Narrative

This document explains the database-focused enhancements implemented in the Travlr project to add soft delete and full auditability across models. The work introduces a reusable Mongoose plugin for audit fields (CreatedAt, UpdatedAt, UpdatedBy) and lifecycle controls (IsDeleted, DeletedAt), and updates controller logic to correctly propagate user context and use soft-delete/restore operations. These changes strengthen data integrity, security, and maintainability.

# What Changed (Summary)

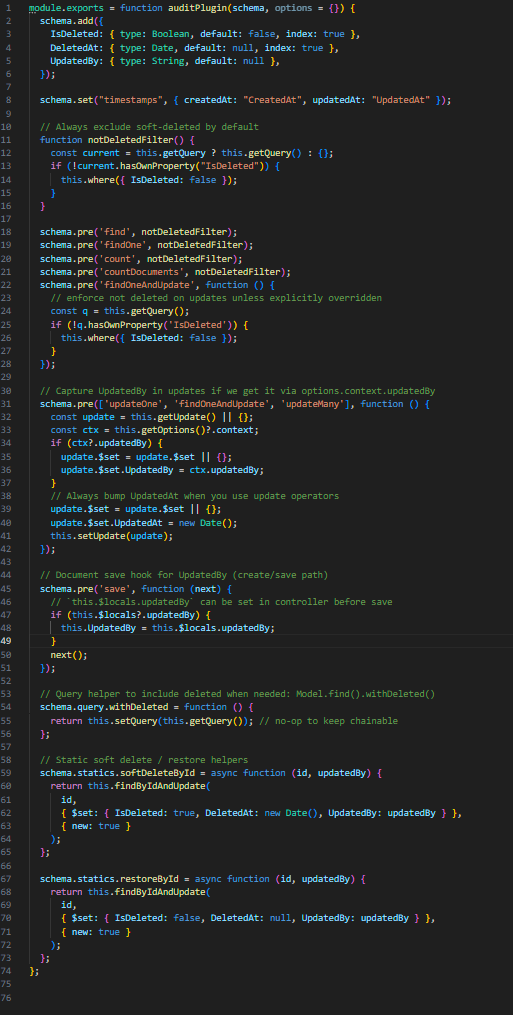
* Added a reusable Mongoose audit plugin (`app\_api/models/plugins/audit.plugin.js`).
* Plugin injects audit fields: CreatedAt, UpdatedAt, UpdatedBy, IsDeleted, DeletedAt.
* Enabled timestamps mapped to custom names (CreatedAt / UpdatedAt).
* Default query behavior excludes soft-deleted records (IsDeleted: false).
* Added pre-hooks to stamp UpdatedAt/UpdatedBy on updates and saves.
* Exposed static helpers: softDeleteById(id, updatedBy) and restoreById(id, updatedBy).
* Updated controller methods to pass `updatedBy` from request context and to call soft delete/restore.
* Kept hard deletes out of the code path; DELETE becomes a soft delete, and a new REST endpoint supports restore.

# Why These Changes Matter

• Compliance & Auditability: Persisting who changed what and when (UpdatedBy, CreatedAt/UpdatedAt) is critical for traceability and compliance. Soft delete preserves history while hiding inactive records by default.  
• Safety & Reversibility: Soft delete prevents accidental data loss and supports record restoration.  
• Maintainability: Centralizing audit logic in a plugin removes duplication and ensures consistent behavior across models.  
• Performance: Indexes on IsDeleted and common query fields allow default filters to remain fast in large collections.

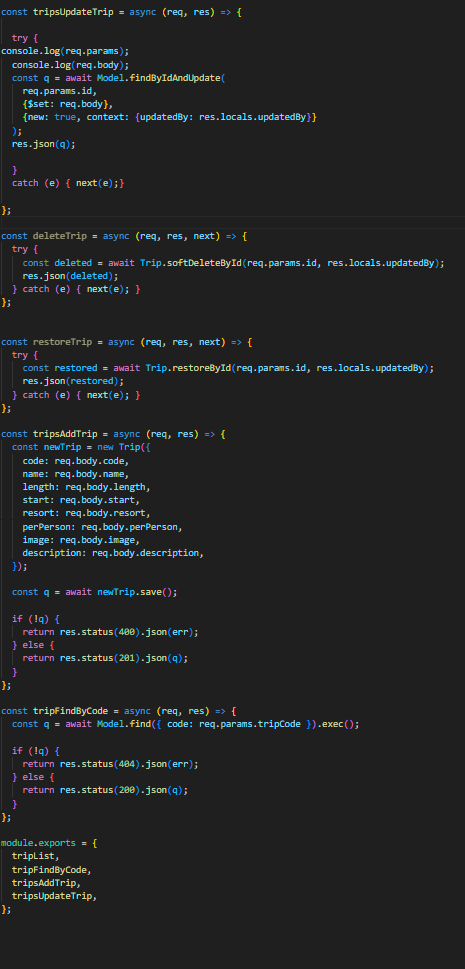
# Audit Plugin – Core Logic

The plugin adds audit/soft-delete fields, sets custom timestamps, excludes deleted documents by default via query hooks, and stamps UpdatedBy/UpdatedAt automatically on updates and saves. It also exposes static helpers for soft delete and restore.



# Controller Updates – Using the Plugin

Controllers now pass the acting user to the model so UpdatedBy is captured. Update operations set context for the plugin; delete/restore use the provided static helpers. Creates populate audit metadata on save via $locals.updatedBy.



# How It Works (Technical Details)

1) Audit Fields & Timestamps: The plugin adds IsDeleted (Boolean), DeletedAt (Date), UpdatedBy (String) and enables Mongoose timestamps mapped to CreatedAt/UpdatedAt.  
2) Default Filtering: Pre-hooks on find/findOne/count ensure queries include { IsDeleted: false } unless explicitly overridden.  
3) UpdatedBy Propagation: For update queries, the plugin reads options.context.updatedBy; for document saves it reads doc.$locals.updatedBy.  
4) Helpers: softDeleteById and restoreById toggle IsDeleted/DeletedAt and stamp UpdatedBy.  
5) Opt-in Access to Deleted: Use .withDeleted() in queries when auditing or reporting requires deleted rows.

# API and Usage Examples

• Create: set doc.$locals.updatedBy before save; plugin stamps UpdatedBy.  
• Update: pass { context: { updatedBy: userIdOrEmail } } to findOneAndUpdate/updateOne.  
• Soft Delete: Trip.softDeleteById(id, updatedBy).  
• Restore: Trip.restoreById(id, updatedBy).  
• Include Deleted: Trip.find(query).withDeleted().

# Migration/Backfill Plan

For existing documents, run a one-time backfill to initialize fields:  
updateMany({ IsDeleted: { $exists: false } }, { $set: { IsDeleted: false, DeletedAt: null } })  
Optionally add indexes: { IsDeleted: 1 }, and compound indexes with business fields (e.g., destination/date).

# Trade-offs and Mitigations

• More Storage per Document: Extra fields increase size slightly; acceptable for auditability gains.  
• Query Complexity: Default filters may surprise new devs; documented and centralized via plugin.  
• Restore Semantics: Business rules should define when a record can be restored; enforced at controller/service layer if needed.

# Program Outcomes Demonstrated

• Software Engineering/Databases: Applied an extensible, reusable plugin to implement cross-cutting concerns cleanly.  
• Security Mindset: Prevented destructive deletes; preserved history and provenance (UpdatedBy).  
• Professional Communication: Documented behavior and exposed simple helpers to reduce team error.