**A red and white logo

AI-generated content may be incorrect.**

**Sede Regional Brunca**

**Campus Pérez Zeledón**

**Análisis de caso practico**

**Curso: Programación 3**

**Profesor: Rubén Mora Vargas**

**II Ciclo-NRC 50248**

**Estudiantes:**

**Jordy Estrada Ortega**

**Andy Hernández Ruiz**

**Harold Ávila Hernández**

**22 de setiembre del 2025**

**Resumen Ejecutivo**

Este proyecto consiste en el desarrollo de una API GraphQL en Java con Spring Boot que funciona como un gestor de hábitos y rutinas saludables. La aplicación está diseñada para permitir a los usuarios crear, personalizar y dar seguimiento a rutinas enfocadas en la salud física, mental, alimentación y sueño. El backend está implementado con tecnologías modernas como Spring Boot, Lombok, Spring Security con JWT, GraphQL, MariaDB y Docker, siguiendo una arquitectura en capas que incluye entidades, DTOs, servicios, repositorios y resolvers.

Tecnologías usadas

* Spring Boot (backend), GraphQL (schema y resolvers), JPA/Hibernate + MariaDB (3FN), Spring Security + JWT (auth/roles), Lombok (menos boilerplate), Docker (app + DB), Log4j2 (logging), pruebas unitarias.
* Todo el pipeline apunta a consultas paginadas y dataset de hasta ~500k filas para pruebas.
* Además se incorporo Bucket4j para rate limiting (control de intentos), un filtro de sanitización de entradas (validación de parámetros y body), y healthchecks en Docker (DB y API) con usuario no-root. Todo esto se configuró sobre Java 21 y MariaDB 11.1 en contenedores.

**Arquitectura**

Capas claras: **entities ↔ repositories (DAO) ↔ services ↔ resolvers GraphQL**, con **DTOs** para no exponer el modelo de dominio directo.

**Patrones aplicados:**

* **DTO (Data Transfer Object):** separa dominio de lo que expongo por GraphQL.
* **Repository (DAO):** persistencia aislada en JpaRepository + paginación/consultas.
* **Service Layer:** reglas de negocio (validaciones, permisos, reseteo de password).
* **Mapper (MapperFor\*):** Entity ⇄ DTO limpio y mantenible.
* **Singleton (config/beans):** JWT provider, servicios y configuración.
* **Factory/Provider (JWT):** creación/validación de tokens en un punto.
* **Policy/Strategy (autorización modular):** verificación **CONSULT/MUTATE** por módulo (RUTINAS, HÁBITOS, etc.) como políticas reutilizables (no ifs sueltos).

La seguridad va por módulo (RUTINAS, HÁBITOS, GUÍAS, PROGRESO, RECORDATORIOS, USUARIOS). Los permisos se representan como CONSULT (consulta) y MUTATE (mutación) por módulo.

* El primer usuario sigue teniendo permisos totales sobre todos los módulos.
* Se puede registrar un usuario normal sin necesidad de token (self-service). A este usuario se le asignan permisos iniciales específicos sobre los módulos ROUTINES, REMINDERS y HABITS (mínimos para que pueda trabajar su rutina personal). Después, un usuario con permisos completos puede ampliar o ajustar esos permisos por módulo.

Capas extra de seguridad:

*  **InputSanitizationFilter + Utils:** si alguien intenta mandar cosas raras (por ejemplo, <script, drop table, etc.) en parámetros o en el body, **lo bloqueo antes** de que entre a mi lógica. También controlo **longitudes** para no aceptar textos absurdamente largos.
*  **RateLimitFilter (Bucket4j):** limito los **intentos de login** por IP (por ejemplo, 5 en 5 minutos). Si alguien fuerza, devuelvo **HTTP 429**.
*  **Docker hardening:** la API corre con **usuario no-root** y healthchecks activos; la DB (MariaDB 11.1) también tiene **healthcheck** y datos persistentes en volumen.

Uso de IA

Generamos esqueletos de DTOs/servicios/resolvers, pom.xml y docker-compose inicial, también borradores de schema.graphqls (queries/mutations por módulo) y textos base del resumen y plan, guía para JWT y el modelo user\_module\_permission (CONSULT/MUTATE) más el primer usuario con permisos completos. Apoyo en el seeder y en la selección de índices de completedactivity.

La IA se usó como acelerador; todo fue revisado y adaptado a los requisitos del curso.

Funcionalidades principales

La API implementa las siguientes funcionalidades:

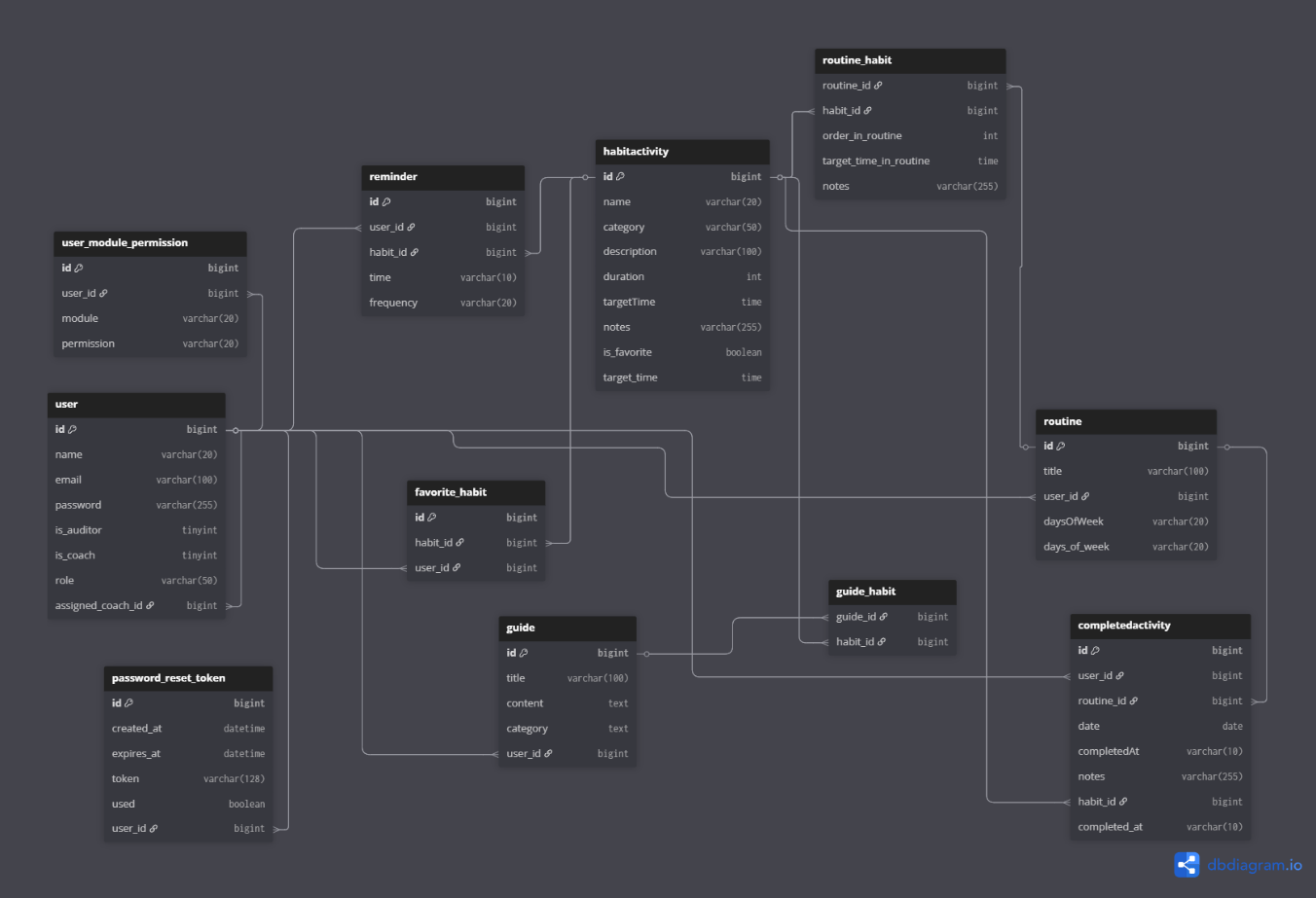
1. Lista de Hábitos por Categoría: el sistema muestra los hábitos físicos, mentales, de sueño y de alimentación.
2. Detalle de Hábito: el usuario puede consultar la información completa de un hábito específico.
3. Lista de Rutinas del Usuario: se listan todas las rutinas creadas por el usuario autenticado.
4. Detalle de Rutina: muestra las actividades, días y hábitos asociados a una rutina.
5. Crear Nueva Rutina: permite definir actividades, tiempos, notas y frecuencia.
6. Seguimiento Diario: registro de actividades completadas en el día.
7. Progreso Semanal: visualización gráfica del cumplimiento de rutinas semanales.
8. Guías Recomendadas: filtradas por categoría o perfil del usuario.
9. Detalle de Guía: despliegue completo de la guía seleccionada.
10. Recordatorios Activos: listado de recordatorios configurados por usuario.
11. Crear Recordatorio: configuración de frecuencia y horario de recordatorios.
12. Favoritos del Usuario: hábitos marcados como favoritos.
13. Login y Registro: autenticación de usuarios, recuperación de contraseña y registro de cuentas nuevas.
14. Lista de Usuarios (modo Coach): vista especial para coaches con sus usuarios asignados.
15. Estadísticas Mensuales: visualización de datos agregados del progreso mensual por categoría.

En conjunto, estas funcionalidades permiten cubrir el ciclo completo de creación, personalización, seguimiento y control de rutinas saludables.

**Plan de Desarrollo**

La tarea se desarrolló en fases claras que se pueden resumir en los siguientes pasos:

1. Inicialización del proyecto: creación del proyecto con Spring Boot, configuración del pom.xml, dependencias de GraphQL, Lombok, JPA, Security y MariaDB.
2. Modelado de base de datos y entidades: diseño en 3FN, creación de entidades y repositorios con JPA.
3. Capa de servicios y DTOs: implementación de DTOs para transferir datos y servicios para la lógica de negocio.
4. GraphQL: construcción de esquemas y resolvers que exponen queries y mutations al cliente.
5. Seguridad con JWT y módulos: configuración de autenticación JWT, creación del primer usuario con permisos completos y control de permisos modulares.
6. Recuperación de contraseña y validaciones: implementación del flujo de recuperación con token temporal y validaciones de entrada.
7. Scripts masivos y paginación: generación de hasta 500 mil registros de prueba y validación de consultas paginadas.
8. Dockerización y pruebas unitarias: creación de contenedores para la aplicación y la base de datos, más cobertura completa de pruebas.

Diagrama de base de datos:

Modelo de datos

user  
id, name, email(unique), password(hash), is\_auditor, is\_coach, role, assigned\_coach\_id(FK→user.id); índice por (name,id) y por assigned\_coach\_id; *check* impide ser auditor y coach a la vez.

user\_module\_permission  
id, user\_id(FK→user), module ENUM('GUIDES','HABITS','PROGRESS','REMINDERS','ROUTINES','USERS'), permission ENUM('CONSULT','MUTATE'); único (user\_id,module).  
→ Aquí viven los permisos por módulo. El primer usuario obtiene todos; desde su JWT se crean/otorgan permisos a otros.

habitactivity  
Catálogo de hábitos: name, category, description, duration, target\_time, notes. (La marca de favorito per-user se maneja en favorite\_habit.)

favorite\_habit  
N:M usuario ↔ hábito favorito; único (user\_id,habit\_id); FK a user y habitactivity.

routine  
title, user\_id(FK→user), days\_of\_week; índice (user\_id,id) para listar “mis rutinas”.

routine\_habit  
Puente N:M rutina ↔ hábito con orden y target\_time\_in\_routine; PK (routine\_id,habit\_id); índices por routine\_id, habit\_id y (routine\_id,order\_in\_routine,habit\_id); ON DELETE CASCADE.

completedactivity  
Registro de progreso: user\_id, routine\_id, habit\_id, date, completed\_at, notes. Índices para escenarios masivos:

* (date,id) y (user\_id,date,id) → “mi progreso” (día/semana).
* (habit\_id,date,id) y (routine\_id,date,id) → agregados por hábito/rutina.  
  FK a user, routine, habitactivity.

reminder  
Recordatorios por usuario/hábito: time, frequency; índices (user\_id,id) y (habit\_id,id).

guide  
title, content, category, user\_id(FK→user); índice (user\_id,id) para listar por autor.

guide\_habit  
Puente N:M guía ↔ hábito para recomendaciones.

password\_reset\_token  
token(unique), created\_at, expires\_at, used, user\_id(FK→user); índice (user\_id,expires\_at).  
→ Recuperación de contraseña con token de un solo uso y expiración.

Con este modelo cumplimos: roles por módulo + auditor, JWT, 3FN, y consultas grandes con paginación e índices pensados para completedactivity.