

Trabajo FINAL DE OpTATIVA NODEJS

GESTIÓN DE HORARIO DOCENTE



**LA HABANA, 2025**

**TUTOR:** ING. REINIER FERNÁNDEZ COELLO

**UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS**

AUTOR:MARIO RIVERO GARCIA

GRUPO 3304

**FACULTAD DE INFORMÁTICA ORGANIZACIONAL**



March 27, 2025

marioriverogarcia13@gmail.com

# RESUMEN

El presente trabajo de investigación desarrolló una solución tecnológica para optimizar la gestión de horarios docentes en la Universidad de las Ciencias Informáticas (UCI), donde actualmente este proceso se realiza de manera manual mediante planillas Excel y su distribución a través de aplicaciones de mensajería instantánea. El estudio partió del análisis de las limitaciones del sistema actual, caracterizado por su baja eficiencia, propensión a errores humanos, dificultades en la colaboración y falta de accesibilidad en tiempo real.

La investigación se fundamentó en el diseño e implementación de una API RESTful desarrollada con Node.js, que integra tecnologías modernas para garantizar escalabilidad, seguridad y robustez. La solución propuesta emplea Express.js como framework backend, utiliza JSON Web Tokens (JWT) para la autenticación segura de usuarios, y se apoya en PostgreSQL como sistema gestor de bases de datos relacional. Todo ello configurando una arquitectura tecnológica óptima para resolver los problemas identificados.

**Palabras clave**: Gestión académica, automatización de horarios, API RESTful, Node.js, transformación digital.

ÍNDICE

[RESUMEN 1](#_Toc193990133)

[INTRODUCCIÓN 4](#_Toc193990134)

[CAPÍTULO 1: TECNOLOGÍAS EMPLEADAS EN EL DESARROLLO DE LA API 6](#_Toc193990135)

[Introducción 6](#_Toc193990136)

[1.1 Node.js 6](#_Toc193990137)

[1.1.1 Características Generales 6](#_Toc193990138)

[1.1.2 Ventajas en el Contexto del Proyecto 6](#_Toc193990139)

[1.2 TypeORM 7](#_Toc193990140)

[1.2.1 Definición y Propósito 7](#_Toc193990141)

[1.2.2. Beneficios para el Sistema 7](#_Toc193990142)

[1.3 JSON Web Tokens (JWT) 7](#_Toc193990143)

[1.3.1 Concepto y Funcionamiento 7](#_Toc193990144)

[1.3.2 Aplicación en el Proyecto 7](#_Toc193990145)

[1.4 PostgreSQL 8](#_Toc193990146)

[1.4.1 Descripción General 8](#_Toc193990147)

[1.4.2 Razones para su Elección 8](#_Toc193990148)

[1.4.3 Integración de Tecnologías 8](#_Toc193990149)

[Capítulo 2: Modelado del Sistema 9](#_Toc193990150)

[Introducción 9](#_Toc193990151)

[1.2. Diagrama de Casos de Uso 9](#_Toc193990152)

[1.3. Diagrama Entidad-Relación 10](#_Toc193990153)

[CONCLUSIONES 11](#_Toc193990154)

# INTRODUCCIÓN

En la actualidad, la gestión de horarios docentes en instituciones universitarias sigue siendo un proceso manual en muchos casos, lo que conlleva ineficiencias, errores y una carga administrativa significativa. En la Universidad de las Ciencias Informáticas (UCI), por ejemplo, los jefes de año son responsables de elaborar los horarios académicos utilizando herramientas como Microsoft Excel, para luego distribuirlos a través de aplicaciones de mensajería como WhatsApp. Este método, aunque funcional, presenta limitaciones en cuanto a escalabilidad, colaboración en tiempo real y accesibilidad centralizada.

La digitalización de procesos administrativos es una tendencia creciente en el ámbito educativo, y las APIs (Interfaces de Programación de Aplicaciones) se han convertido en una solución eficiente para automatizar tareas repetitivas y mejorar la gestión de datos. Node.js, gracias a su escalabilidad y capacidad para manejar operaciones en tiempo real, se posiciona como una tecnología idónea para desarrollar una solución que optimice la creación, edición y distribución de horarios docentes.

**Problema de Investigación**

El principal problema identificado es la falta de un sistema centralizado y automatizado para la gestión de horarios docentes en la UCI, lo que obliga a los jefes de año a realizar esta labor de manera manual mediante planillas Excel, con los siguientes inconvenientes:

-Dificultad en la colaboración: Los archivos se envían por WhatsApp o correo electrónico, lo que puede generar versiones desactualizadas y falta de sincronización.

-Propensión a errores humanos: La manipulación manual de datos aumenta el riesgo de inconsistencias en los horarios.

-Falta de accesibilidad inmediata: Los docentes y estudiantes no cuentan con una plataforma unificada para consultar los horarios en tiempo real.

-Escalabilidad limitada: A medida que crece la cantidad de asignaturas y profesores, el proceso manual se vuelve más complejo e ineficiente.

Ante esta problemática, surge la necesidad de desarrollar una API en Node.js que permita gestionar de manera eficiente los horarios docentes, proporcionando una solución escalable, en tiempo real y de fácil acceso para todos los involucrados. Esta investigación busca demostrar cómo una solución tecnológica puede optimizar un proceso administrativo crítico en el entorno universitario.

**Objetivo General**

Diseñar e implementar una API en Node.js para la gestión automatizada de horarios docentes en la UCI, con el fin de eliminar los procesos manuales basados en Excel y WhatsApp, mejorando la eficiencia, precisión y accesibilidad de la información.

# CAPÍTULO 1: TECNOLOGÍAS EMPLEADAS EN EL DESARROLLO DE LA API

## Introducción

En este capítulo se presentan las tecnologías fundamentales seleccionadas para el desarrollo de la API de gestión de horarios docentes. La elección de estas herramientas se basó en su capacidad para cumplir con los requisitos de escalabilidad, seguridad y eficiencia que demanda el sistema. Las tecnologías principales utilizadas incluyen Node.js como entorno de ejecución, TypeORM para la gestión de datos, JWT para la autenticación y PostgreSQL como sistema de base de datos relacional. A continuación, se describen sus características generales y su relevancia en el proyecto.

## 1.1 Node.js

### 1.1.1 Características Generales

Node.js es un entorno de ejecución para JavaScript construido sobre el motor V8 de Chrome, diseñado para desarrollar aplicaciones escalables y de alto rendimiento. Su arquitectura está basada en un modelo no bloqueante (event-loop), lo que lo hace especialmente eficiente para manejar operaciones de entrada/salida (E/S) asíncronas, como solicitudes HTTP y acceso a bases de datos.

### 1.1.2 Ventajas en el Contexto del Proyecto

Eficiencia en operaciones concurrentes: Ideal para APIs que requieren manejar múltiples peticiones simultáneas.

Ecosistema robusto: Disponibilidad de una amplia variedad de módulos y bibliotecas a través de npm (Node Package Manager).

Flexibilidad: Permite la integración con diversos frameworks y herramientas para backend.

Comunidad activa: Gran cantdad de recursos y soporte disponible para resolver problemas comunes.

## 1.2 TypeORM

### 1.2.1 Definición y Propósito

TypeORM es una herramienta de mapeo objeto-relacional (ORM) que facilita la interacción con bases de datos mediante programación orientada a objetos. Soporta tanto bases de datos SQL como NoSQL y está diseñado para trabajar con TypeScript, aunque también es compatible con JavaScript.

### 1.2.2. Beneficios para el Sistema

Abstracción de la base de datos: Simplifica las operaciones CRUD sin necesidad de escribir consultas SQL manuales.

Soporte para migraciones: Permite gestionar cambios en el esquema de la base de datos de manera controlada y versionada.

Manejo de relaciones entre entidades: Facilita la definición y consulta de relaciones complejas, como las existentes entre horarios, profesores y aulas.

Compatibilidad con múltiples bases de datos: Ofrece flexibilidad para adaptarse a diferentes sistemas de almacenamiento.

## 1.3 JSON Web Tokens (JWT)

### 1.3.1 Concepto y Funcionamiento

JWT es un estándar abierto utilizado para la transmisión segura de información entre partes en forma de tokens compactos y autónomos. Estos tokens están compuestos por tres partes: un encabezado, un payload y una firma digital, lo que garantiza su integridad y autenticidad.

### 1.3.2 Aplicación en el Proyecto

Autenticación sin estado (stateless): Elimina la necesidad de almacenar sesiones en el servidor, reduciendo la carga del sistema.

Seguridad: La firma digital previene la manipulación de los datos contenidos en el token.

Personalización: Permite incluir información adicional (claims) para gestionar permisos y roles de usuarios.

## 1.4 PostgreSQL

### 1.4.1 Descripción General

PostgreSQL es un sistema de gestión de bases de datos relacionales (RDBMS) de código abierto, reconocido por su robustez, escalabilidad y cumplimiento con los principios ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad).

### 1.4.2 Razones para su Elección

Manejo de datos estructurados: Ideal para modelos de datos con relaciones complejas, como los requeridos en la gestión de horarios.

Soporte para consultas avanzadas: Permite ejecutar operaciones complejas con alto rendimiento.

Escalabilidad: Adecuado para adaptarse al crecimiento futuro de la aplicación.

Fiabilidad: Garantiza la integridad de los datos incluso en entornos con alta concurrencia.

### 1.4.3 Integración de Tecnologías

La combinación de estas tecnologías permite construir una API eficiente, segura y fácil de mantener:

Node.js actúa como la base del servidor, gestionando las peticiones y respuestas.

TypeORM se encarga de la comunicación con la base de datos, abstraiendo la complejidad de las consultas SQL.

JWT proporciona un mecanismo seguro para autenticar usuarios y proteger los endpoints.

PostgreSQL almacena y organiza los datos de manera estructurada, garantizando su consistencia.

Esta sinergia asegura que el sistema cumpla con los objetivos de automatización, precisión y accesibilidad planteados en el proyecto.

# Capítulo 2: Modelado del Sistema

## Introducción

En este capítulo se presenta el modelado del sistema mediante dos herramientas fundamentales:

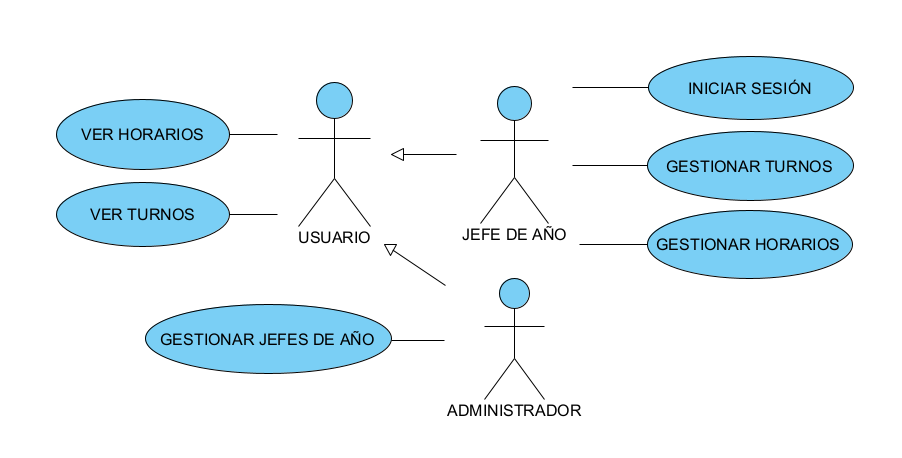
Diagrama Entidad-Relación: Representa la estructura de la base de datos y las relaciones entre las entidades principales.

Diagrama de Casos de Uso: Describe las interacciones entre los actores del sistema y las funcionalidades clave.

Estos modelos permiten visualizar el diseño conceptual antes de su implementación técnica.

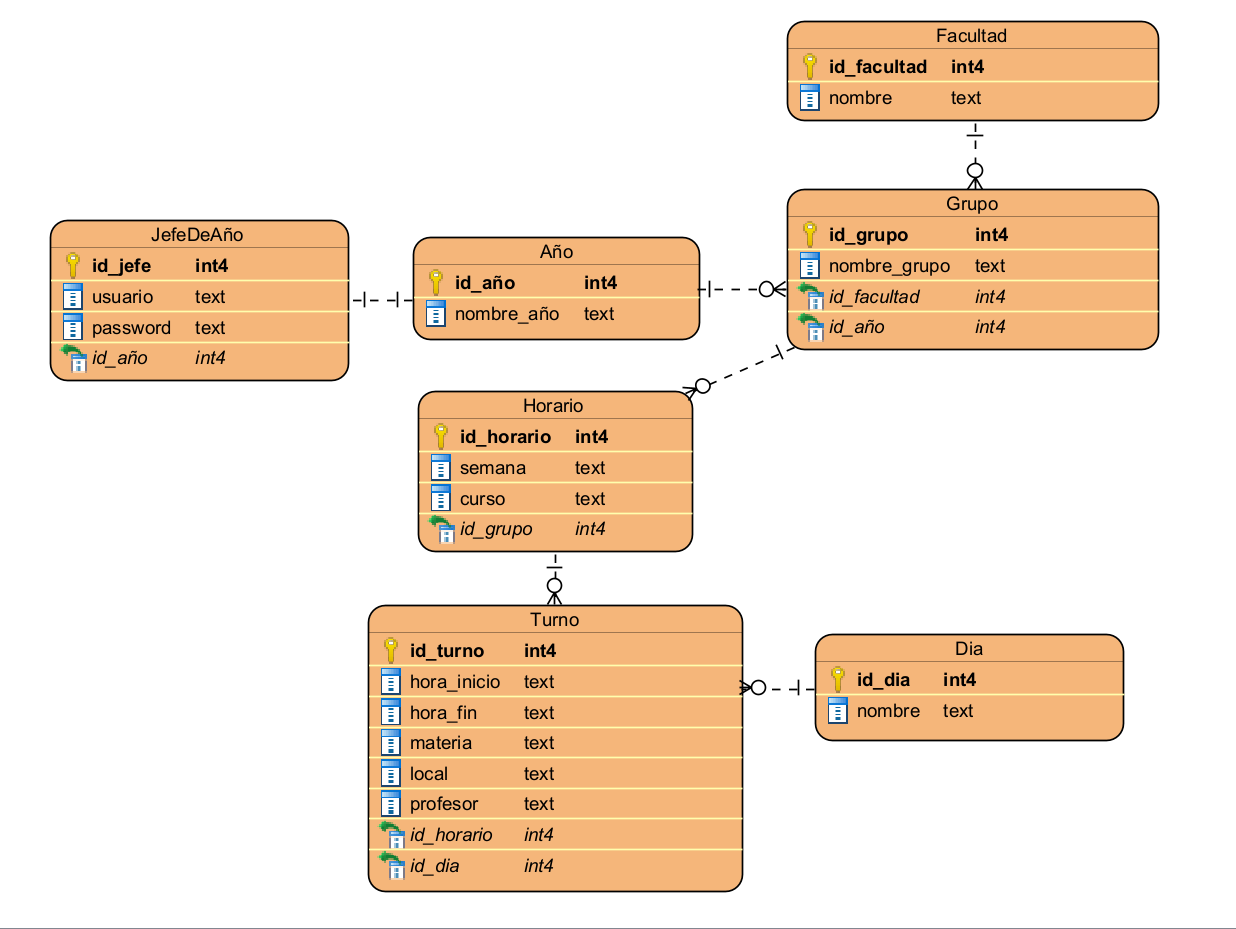
## 1.2. Diagrama de Casos de Uso

El Diagrama de Casos de Uso resume las funcionalidades principales desde la perspectiva de los usuarios:



## 1.3. Diagrama Entidad-Relación

El Diagrama Entidad-Relacion define las entidades principales del sistema y sus relaciones, incluyendo:



# CONCLUSIONES

Los principales resultados obtenidos demuestran que la implementación de esta solución tecnológica permite:

-Automatizar completamente el proceso de creación y gestión de horarios

-Centralizar la información en una plataforma única accesible en tiempo real

-Reducir significativamente los errores asociados al procesamiento manual

-Mejorar la colaboración entre los diferentes actores involucrados

-Garantizar la escalabilidad del sistema ante el crecimiento institucional

La investigación concluye que la adopción de esta solución representa un avance significativo en la digitalización de procesos administrativos académicos, con potencial de replicación en otras instituciones de educación superior.