# Informe Constellation Backend

En este informe se presentará un análisis exhaustivo de las funcionalidades implementadas en las APIs desarrolladas durante la creación del backend del proyecto Constellation. Además, se ofrecerá una explicación detallada sobre la implementación de características clave como autenticación, autorización, persistencia de datos en la base de datos y la ejecución de pruebas.

## - Endpoints

### Gestión de Actividades

Este módulo se encarga de la gestión de las actividades que los estudiantes deben realizar dentro de un curso. Permite a los profesores crear actividades y establecer fechas de entrega, así como actualizarla en caso de ser necesario. Cada actividad tendrá asociada varias actividades que podrán ser definidas por cada profesor con una API explicada más adelante.

| Método | Ruta | Descripción | Parámetros | Respuesta |
|---|---|---|---|---|
| GET | /activities | Obtener todas las actividades | N/A | Devuelve un arreglo de objetos que representan las actividades |
| GET | /activities/:id | Obtener una actividad por ID | `id` (string): El ID de la actividad a buscar. | Devuelve el objeto de la actividad con el ID especificado. |
| POST | /activities | Crear una nueva actividad | `createActivityDto` (objeto): Contiene los datos de la nueva actividad, como el curso al que pertenece y otros detalles. | Devuelve el objeto de la actividad recién creada. |

| | | | | |
|---|---|---|---|---|
| PATCH | /activities/:id | Actualizar una actividad existente | `id` (string): El ID de la actividad a actualizar.<br><br>`updateActivityDto` (objeto): Contiene los datos actualizados de la actividad. | Devuelve la actividad actualizada. |
| DELETE | /activities/:id | Eliminar una actividad | `id` (string): El ID de la actividad a eliminar. | Confirmación de que la actividad ha sido eliminada. |

**Autenticación y Gestión de Usuarios**

Este módulo maneja el registro, inicio de sesión y autorización de los usuarios dentro del sistema. Proporciona funcionalidades para gestionar diferentes tipos de usuarios (estudiantes y profesores). Incluye la lógica para proteger rutas y recursos, asegurando que solo los usuarios autenticados y autorizados puedan acceder a ciertas funcionalidades.

| **Método** | **Ruta** | **Descripción** | **Parámetros** | **Respuesta** |
|---|---|---|---|---|
| POST | /auth | Crear un nuevo usuario | `createUserDto` (objeto): Contiene los datos del usuario a crear, como nombre, email, contraseña, código de usuario y rol. | Devuelve el objeto del usuario recién creado. |
| GET | /auth | Obtener todos los usuarios | N/a | Devuelve un arreglo de objetos que representan a los usuarios registrados. |
| GET | /auth/:id | Obtener un usuario por ID | `id` (string): El ID o el código de usuario para buscar al usuario. | Devuelve el objeto del usuario con el identificador especificado. |

| | | | | |
|---|---|---|---|---|
| PATCH | /auth/:id | Actualizar datos de un usuario | `id` (string): El ID o el código de usuario del usuario a actualizar.<br><br>`updateUserDto` (objeto): Contiene los datos actualizados del usuario. | Devuelve el usuario actualizado. |
| DELETE | /auth/:id | Eliminar un usuario | `id` (string): El ID o el código de usuario del usuario a eliminar. | Confirmación de que el usuario ha sido eliminado. |
| POST | /auth/login | Iniciar sesión de un usuario | `loginUserDto` (objeto): Contiene el email y la contraseña del usuario. | Devuelve un objeto con el `user_id`, `email`, y un token JWT para la sesión. |
| POST | /auth/uploadStudents/:id | Subir archivo de estudiantes en Excel | `file` (archivo): Un archivo Excel que contiene datos de usuarios. | Devuelve un mensaje confirmando la carga y los datos procesados. |

**Gestión de Cursos**

Este módulo permite la creación y administración de cursos en la plataforma. Los profesores pueden crear nuevos cursos, inscribir a estudiantes a partir de un archivo excel y gestionar el contenido del curso. Cada curso tiene sus respectivas actividades asociadas. También se encarga de la organización de la información relacionada con los cursos, como descripciones, horarios y requisitos.

| **Método** | **Ruta** | **Descripción** | **Parámetros** | **Respuesta** |
|---|---|---|---|---|
| POST | /courses | Crear un nuevo curso | `createCourseDto` (objeto): Contiene los datos del curso a crear. | Devuelve el objeto del curso recién creado. |
| GET | /courses | Obtener todos los cursos | N/a | Devuelve un array de objetos con la información de todos los cursos, incluyendo |

| | | | | |
|---|---|---|---|---|
| | | | | las actividades asociadas. |
| GET | /courses/:id | Obtener un curso por ID | `id` (string): El identificador único del curso a buscar. | Devuelve el objeto del curso con el ID especificado. |
| PATCH | /courses/:id | Actualizar un curso | `id` (string): El identificador único del curso a actualizar.<br>`updateCourseDto` (objeto): Contiene los campos a actualizar del curso. | Devuelve el objeto del curso actualizado. |
| DELETE | /courses/:id | Eliminar un curso | `id` (string): El identificador único del curso a eliminar. | Devuelve un mensaje confirmando que el curso ha sido eliminado. |
| POST | /courses/:courseId/user/:userId | Asignar un usuario a un curso | `courseId` (string): El identificador único del curso que se va a asignar.<br>`userId` (string): El identificador único del usuario al que se va a asignar el curso. | Devuelve el objeto relacionado con la asignación del curso al usuario. |
| POST | /courses/teams/:courseId | Crear grupos para un curso | `courseId` (string): El identificador único del curso para el cual se van a crear los grupos. | Devuelve el objeto que contiene la información de los grupos creados. |

**Gestión de Criterios**

Este módulo se encarga de definir y gestionar los criterios de evaluación que se utilizarán para calificar a los estudiantes en actividades a través de las rúbricas. Permite a los profesores establecer criterios específicos, sus descripciones y ponderaciones, lo que proporciona un marco claro para la evaluación.

| Método | Ruta | Descripción | Parámetros | Respuesta |
|---|---|---|---|---|
| POST | /criteria | Crear un nuevo criterio | `criteria` (CreateCriteriaDto): Objeto que contiene la información necesaria para crear el criterio. | Devuelve el criterio creado. |
| GET | /criteria | Obtener todos los criterios | N/a | Devuelve una lista de todos los criterios |
| GET | /criteria/:id | Obtener un criterio por ID | `id` (string): El identificador único del criterio a buscar. | Devuelve el criterio encontrado junto con su rúbrica asociada. |
| PATCH | /criteria/:id | Actualizar un criterio | `id` (string): El identificador único del criterio que se va a actualizar.<br>`updateCriteriaDto` (CreateCriteriaDto): Objeto que contiene los nuevos valores para actualizar el criterio. | Devuelve el criterio actualizado. |
| DELETE | /criteria/:id | Eliminar un criterio | `id` (string): El identificador único del criterio a eliminar. | Devuelve una confirmación de que el criterio ha sido eliminado. |

**Gestión de Notas de Criterios**

Este módulo permite la asignación y gestión de las calificaciones relacionadas con los criterios de evaluación. Permite registrar las notas puestas por los otros estudiantes en cada criterio para un estudiante.

| Método | Ruta | Descripción | Parámetros | Respuesta |
|---|---|---|---|---|
| POST | /criteria-grade | Crear una nueva nota de criterio | `createCriteriaGradeDto` (CreateCriteriaGradeDto): Objeto que contiene | Devuelve la evaluación de criterio creada, incluyendo su ID único y |

| | | | | |
|---|---|---|---|---|
| | | | la información necesaria para crear la evaluación de criterio | las asociaciones correspondientes. |
| GET | /criteria-grade | Obtener todas las notas de criterios | N/a | Devuelve una lista de todas las evaluaciones de criterio, incluyendo su asociación con los criterios. |
| GET | /criteria-grade/:id | Obtener una nota de criterio por ID | `id` (string): El identificador único de la evaluación de criterio a buscar. | Devuelve la evaluación de criterio encontrada junto con el criterio asociado. |
| PATCH | /criteria-grade/:id | Actualizar una nota de criterio | `id` (string): El identificador único de la evaluación de criterio que se va a actualizar.<br>`updateCriteriaGradeDto` (UpdateCriteriaGradeDto): Objeto que contiene los nuevos valores. | Devuelve la evaluación de criterio actualizada. |
| DELETE | /criteria-grade/:id | Eliminar una nota de criterio | `id` (string): El identificador único de la evaluación de criterio a eliminar. | Devuelve una confirmación de que la evaluación de criterio ha sido eliminada. |

**Gestión de Rúbricas**

Este módulo se encarga de la creación y gestión de rúbricas que sirven como herramientas de evaluación para las actividades. Permite a los profesores diseñar rúbricas, cada una con sus criterios asociados, facilitando una evaluación más objetiva y estructurada.

| Método | Ruta | Descripción | Parámetros | Respuesta |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| POST | /rubric | Crear una nueva rúbrica | `createRubricDto` (CreateRubricDto): Objeto que contiene la información necesaria para crear la rúbrica | Devuelve la rúbrica creada, incluyendo su ID único y la asociación con la actividad correspondiente. |
| GET | /rubric | Obtener todas las rúbricas | N/a | Devuelve una lista de todas las rúbricas, incluyendo sus criterios y la actividad asociada. |
| GET | /rubric/:id | Obtener una rúbrica por ID | `id` (string): El identificador único de la rúbrica a buscar. | Devuelve la rúbrica encontrada junto con la actividad y los criterios asociados. |
| PATCH | /rubric/:id | Actualizar una rúbrica | `id` (string): El identificador único de la rúbrica que se va a actualizar.<br>`updateRubricDto` (UpdateRubricDto): Objeto que contiene los nuevos valores | Devuelve la rúbrica actualizada. |
| DELETE | /rubric/:id | Eliminar una rúbrica | `id` (string): El identificador único de la rúbrica a eliminar. | Devuelve una confirmación de que la rúbrica ha sido eliminada. |

**Gestión de Notas de Rúbricas**

Este módulo permite la asignación y gestión de las calificaciones derivadas de las rúbricas utilizadas en la evaluación de actividades. Facilita calcular las notas finales de los estudiantes basándose en las evaluaciones de los diferentes criterios establecidos en las rúbricas.

| Método | Ruta | Descripción | Parámetros | Respuesta |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| POST | /rubric-grade | Crear una nueva nota de rúbrica | `createRubricGradeDto` (CreateRubricGradeDto): Objeto que contiene la información necesaria para crear la calificación | Devuelve la calificación de rúbrica creada. |
| GET | /rubric-grade | Obtener todas las notas de rúbricas | N/a | Devuelve una lista de todas las calificaciones de rúbrica. |
| GET | /rubric-grade/:id | Obtener una nota de rúbrica por ID | `id` (string): El identificador único de la calificación de rúbrica a buscar. | Devuelve la calificación de rúbrica encontrada junto con la rúbrica asociada. |
| PATCH | /rubric-grade/:id | Actualizar una nota de rúbrica | `id` (string): El identificador único de la calificación de rúbrica que se va a actualizar.<br>`updateRubricGradeDto` (UpdateRubricGradeDto): Objeto que contiene los nuevos valores | Devuelve la calificación de rúbrica actualizada. |
| DELETE | /rubric-grade/:id | Eliminar una nota de rúbrica | `id` (string): El identificador único de la calificación de rúbrica a eliminar. | Devuelve una confirmación de que la calificación ha sido eliminada. |

**Gestión de Equipos**

Este módulo se ocupa de la formación y administración de equipos de trabajo dentro de la plataforma. Permite a los profesores crear equipos, los cuales se crearán según las habilidades y horario disponible de cada estudiante.

| Método | Ruta | Descripción | Parámetros | Permisos |
|---|---|---|---|---|
| POST | /team | Crear un nuevo equipo | `createTeamsDto` (CreateTeamsDto): Objeto que contiene la información necesaria para crear el equipo. | Devuelve el equipo creado |
| GET | /team | Obtener todos los equipos | N/a | Devuelve una lista de todos los equipos. |
| GET | /team/:id | Obtener un equipo por ID | `id` (string): El identificador único del equipo a buscar. | Devuelve el equipo encontrado. |
| PATCH | /team/:id | Actualizar un equipo | `id` (string): El identificador único del equipo que se va a actualizar.<br>`updateTeamsDto` (UpdateTeamsDto): Objeto que contiene los nuevos valores | Devuelve el equipo actualizado. |
| DELETE | /team/:id | Eliminar un equipo | `id` (string): El identificador único del equipo a eliminar. | Devuelve una confirmación de que el equipo ha sido eliminado. |
| POST | /team/:teamId/user/:userId | Asignar un usuario a un equipo | `teamId` (string): El identificador único del equipo al que se va a agregar el usuario.<br>`userId` (string): El identificador único del usuario que se va a agregar al equipo. | Devuelve el usuario actualizado con el equipo agregado. |

**Gestión de Horarios**

Este módulo se encarga de la gestión de los horarios que los estudiantes pueden elegir para estar disponibles para realizar trabajos o adelantos de talleres. La finalidad de este sistema es facilitar la formación de grupos con horarios similares, optimizando así la colaboración entre estudiantes en sus actividades académicas.

| Método | Ruta | Descripción | Parámetros | Respuesta |
|---|---|---|---|---|
| GET | /schedule | Obtener todos los horarios disponibles | N/A | Devuelve un arreglo de objetos que representan los horarios disponibles. |
| GET | /schedule/:id | Obtener un horario por ID | id (string): El ID del horario a buscar. | Devuelve el objeto del horario con el ID especificado. |
| POST | /schedule | Crear un nuevo horario disponible | createScheduleDto (objeto): Contiene los datos del nuevo horario, como el curso y otros detalles. | Devuelve el objeto del horario recién creado. |
| PATCH | /schedule/:id | Actualizar un horario existente | id (string): El ID del horario a actualizar. updateScheduleDto (objeto): Contiene los datos actualizados del horario. | Devuelve el horario actualizado. |
| DELETE | /schedule/:id | Eliminar un horario | id (string): El ID del horario a eliminar. | Confirmación de que el horario ha sido eliminado. |

**Gestión de Habilidades**

Este módulo se encarga de la gestión de las habilidades técnicas que los estudiantes pueden elegir, tales como front end, backend, entre otras. El objetivo es que los estudiantes indiquen qué habilidades poseen, lo que permitirá formar equipos de trabajo con una variedad adecuada de competencias.

| Método | Ruta | Descripción | Parámetros | Respuesta |
|---|---|---|---|---|
| GET | /skills | Obtener todas las habilidades disponibles | N/A | Devuelve un arreglo de objetos que representan las habilidades. |
| GET | /skills/:id | Obtener una habilidad por ID | id (string): El ID de la habilidad a buscar. | Devuelve el objeto de la habilidad con el ID especificado. |
| POST | /skills | Crear una nueva habilidad | createSkillDto (objeto): Contiene los datos de la nueva habilidad. | Devuelve el objeto de la habilidad recién creada. |
| PATCH | /skills/:id | Actualizar una habilidad existente | id (string): El ID de la habilidad a actualizar.<br><br>updateSkillDto (objeto): Contiene los datos actualizados de la habilidad. | Devuelve la habilidad actualizada. |

| | | | | |
|---|---|---|---|---|
| DELETE | /skills/:id | Eliminar una habilidad | id (string): El ID de la habilidad a eliminar. | Confirmación de que la habilidad ha sido eliminada. |
| POST | /skills/:skillId/user/:userId | Agregar una habilidad a un estudiante | skillId (string): El ID de la habilidad.<br><br>userId (string): El ID del estudiante. | Devuelve la confirmación de que la habilidad ha sido agregada al estudiante. |

## - Autenticación

La autenticación se implementa para verificar la identidad de los usuarios que intentan acceder al sistema. Se basa en los siguientes conceptos:

1. **Módulo de Autenticación y Gestión de Usuarios**: Este módulo gestiona el proceso de registro, inicio de sesión y recuperación de contraseñas para los usuarios. Utiliza servicios como el **AuthService**, que se encarga de las operaciones de autenticación y gestión de usuarios.
2. **Estrategia de Autenticación y Tokens (JWT)**: Se utiliza la estrategia de JWT (JSON Web Tokens) para autenticar a los usuarios. Cuando un usuario inicia sesión con sus credenciales, el sistema genera un token JWT que se envía al cliente. Este token se incluye en las solicitudes futuras para autenticar al usuario sin que tenga que volver a ingresar sus credenciales.
3. **Integración con NestJS**: La autenticación está integrada en el framework utilizando decoradores personalizados, como **@Auth()**, que protegen las rutas y aseguran que solo los usuarios autenticados puedan acceder a ciertas funcionalidades. Además, se emplea **ValidRoles** para definir roles específicos, como **teacher**, lo que permite controlar qué acciones pueden realizar los diferentes tipos de usuarios.

## - Autorización

La autorización determina qué recursos y acciones están disponibles para los usuarios autenticados, dependiendo de sus roles y permisos. Esto se implementa de la siguiente manera:

1. **Roles y Permisos:** La aplicación define roles específicos, como **teacher** y **student**, para controlar el acceso a ciertas funcionalidades. Utilizando el decorador **@Auth(ValidRoles.teacher)** en los controladores, se restringe el acceso únicamente a los usuarios que posean el rol adecuado.
2. **Decoradores Personalizados:** Se emplean decoradores personalizados como **@Auth()** para aplicar la lógica de autorización en las rutas de los controladores. Estos decoradores validan el token JWT del usuario, verifican su rol y aseguran que cuente con los permisos necesarios para acceder a las funcionalidades protegidas.
3. **Manejo de Errores de Autorización:** Si un usuario intenta acceder a un recurso sin los permisos adecuados, el sistema lanza una excepción, evitando así el acceso a rutas protegidas. Este manejo de errores garantiza que solo los usuarios autorizados puedan interactuar con las partes sensibles de la aplicación.

## - Persistencia en la Base de Datos

La persistencia de datos se gestiona a través de TypeORM, integrado en NestJS, utilizando una base de datos relacional como PostgreSQL. A continuación se destacan los aspectos clave:

1. **Entidades y Repositorios:** Cada módulo tiene entidades que representan tablas en la base de datos, definidas con decoradores de TypeORM, como **@Entity()** y **@Column()**. Esto permite estructurar los datos adecuadamente.
2. **Relaciones Entre Entidades:** Las relaciones entre tablas, como ManyToOne y OneToMany, se gestionan explícitamente en las entidades, lo que permite consultas eficientes y mantiene la integridad de los datos.
3. **Inyección de Repositorios:** Los servicios utilizan inyección de dependencias para acceder a los repositorios de TypeORM. Por ejemplo, **RubricGradeService** usa **@InjectRepository(RubricGrade)** para realizar operaciones relacionadas con las notas.
4. T**ransacciones y Consistencia de Datos:** Se asegura la consistencia de las operaciones mediante el uso de métodos asíncronos con await, permitiendo que las actualizaciones y eliminaciones sean atómicas.

5. **Manipulación de Datos:** Los métodos de servicio, como create, update y remove, manejan las operaciones CRUD sobre las entidades, interactuando directamente con la base de datos a través de los repositorios.

## - Ejecución de Pruebas

La implementación de pruebas en el proyecto se llevó a cabo utilizando diferentes tipos de archivos de especificación, que permiten verificar el correcto funcionamiento de los componentes de la aplicación. A continuación se describen los principales tipos de pruebas y su propósito:

**Pruebas de Controlador (controller.spec.ts)**:

**Propósito**: Estas pruebas se centran en verificar que los controladores llamen correctamente a los métodos de los servicios y manejen adecuadamente las entradas. Se asegura que los controladores procesen correctamente las solicitudes y devuelvan las respuestas esperadas en función de los resultados de los servicios.

**Ejecución**: npm run test

**Pruebas de Servicio (service.spec.ts)**:

**Propósito**: Estas pruebas evalúan la lógica de negocio contenida en los servicios. Se verifican los métodos de los servicios para asegurar que realicen las operaciones esperadas, como crear, actualizar o eliminar datos.

**Ejecución**: npm run test

**Pruebas de End to End (e2e-spec.ts)**:

**Propósito**: Estas pruebas verifican el comportamiento de la aplicación en su totalidad, simulando interacciones del usuario desde el inicio de sesión hasta la realización de acciones en la interfaz. Esto asegura que todos los componentes trabajen juntos correctamente.

**Ejecución**:  npm run test:e2e