Auth

Login

Description: Login a user. User roles are admin, student or teacher. Upon successful login the user is returned.

Method: POST

Route: /auth/login

Errors: "Falta email, contraseña o rol de usuario", "Rol de usuario no existe", "Usuario no existe", "Contraseña incorrecta", "Un error ocurrió al intentar hacer login"

Example:

```
URL: {server url}/api/auth/login
Body:
{
    "email": "juan@gmail.com",
    "password": "juan",
    "role": "student"
}
Returns
{
    "_id": "611369eba56e672ba8fe781b",
    "id": "206754833",
    "email": "juan@gmail.com",
    "password": "$2a$10$uM7P5ynobi7Bwd1NHVsv1ehr46Mn9z6j7eEfilYck5ZQN5Jol.2.C",
    "name": "Juan",
    "lastname": "Rojas Vargas",
    "grade": 6,
    "__v": 0
}
```

Change Password

Description: Change an user's password. User roles are admin, student or teacher.

Method: PUT

Route: /auth/password

Errors: "Falta un campo de cambio de contraseña", "Rol de usuario no existe", "Usuario no existe", "Contraseña actual incorrecta", "Un error ocurrió al intentar cambiar la contraseña"

Example:

```
URL: {server url}/api/auth/passwords
Body:
{
    "email": "mepdigitalweb.admi@gmail.com",
    "role": "admin",
    "currentPassword": "m9K481uG",
    "newPassword": "pass123"
}
Returns
{
    "message": "Se cambió la contraseña exitosamente"
}
```

Admins

Create Admin

Description: Create admin
Method: POST
Route: /admins
Errors: "Falta un campo del admin", "Correo ya está registrado", "Un error ocurrió enviando el correo con la contraseña", "Un error ocurrió creando el admin"
Example:

```
URL: {server url}/api/admins
Body:
{
    "email": "mepdigitalweb.admi@gmail.com"
}
Returns
{
    "message": "Se creó el admin exitosamente"
}
```

Delete Admin

Description: Delete admin
Method: DELETE
Route: /admins
Errors: "Admin no existe", "Un error ocurrió eliminando el admin"
Example:

```
URL: {server url}/api/admins
Body:
{
    "email": "admin2@gmail.com"
}
Returns
{
   "message": "Se eliminó el admin exitosamente"
}
```

Students

Create Student

Description: Create student
Method: POST
Route: /students
Errors: "Falta un campo del estudiante", "Carnet ya está registrado", "Correo ya está registrado", "Grado inválido", "Un error ocurrió enviando el correo con la contraseña", "Un error ocurrió creando el estudiante"
Example:

```
URL: {server url}/api/students
Body: {
   "id": "2088654382",
   "email": "armando@gmail.com",
   "name": "Armando",
   "lastname": "Arias Chacon",
   "grade": 4
}
Returns
{
   "message": "Se creó el estudiante exitosamente"
}
```

Get Student

Description: Get student by id
Method: GET
Route: /students/id
Errors: "Estudiante no existe", "Un error ocurrió recuperando el estudiante"
Example:

```
URL: {server url}/api/students/206754833
Body: { }
Returns
{
   "student": {
      "_id": "611369eba56e672ba8fe781b",
      "id": "206754833",
      "email": "juan@gmail.com",
      "password": "$2a$10$uM7P5ynobi7Bwd1NHVsv1ehr46Mn9z6j7eEfilYck5ZQN5Jol.2.C",
      "name": "Juan",
      "lastname": "Rojas Vargas",
      "grade": 6,
      "__v": 0
   }
}
```

Get All Students

Description: Get all students
Method: GET
Route: /students
Errors: "Un error ocurrió recuperando todos los estudiantes"
Example:

```
URL: {server url}/api/students
Body: { }
Returns
{
   "students": [
      {
         "_id": "611369eba56e672ba8fe781b",
         "id": "206754833",
         "email": "juan@gmail.com",
         "password": "$2a$10$uM7P5ynobi7Bwd1NHVsv1ehr46Mn9z6j7eEfilYck5ZQN5Jol.2.C",
         "name": "Juan",
         "lastname": "Rojas Vargas",
         "grade": 6,
         "__v": 0
      },
      {
         "_id": "61145b3f5d7aec3ca0f3bc68",
         "id": "205746383",
         "email": "oscar@gmail.com",
         "password": "$2a$10$zpANhd7JPdTyc4n8cY5h1e0kqCTvhNAPI3rmwK/6C2MGY/URDkzrO",
         "name": "Oscar",
         "lastname": "Mendez Yoses",
         "grade": 3,
         "__v": 0
      }
   ]
}
```

Update Student

Description: Update student by id.
Method: PUT
Route: /students/id
Errors: "Falta un campo del estudiante", "Correo ya está en uso", "Grado inválido", "Estudiante no existe", “Estudiante no puede cambiar de grado porque está inscrito en cursos”, "Un error ocurrió actualizando el estudiante"
Example:

```
URL: {server url}/api/students/2088654382
Body:
{
   "email": "armando@gmail.com",
   "name": "Armando",
   "lastname": "Arias Chacon",
   "grade": 4
}
Returns
{
   "message": "Se actualizó el estudiante exitosamente"
}
```

Delete Student

Description: Delete student by id. The student is removed from all enrolled courses.
Method: DELETE
Route: /students/id
Errors: "Estudiante no existe", "Un error ocurrió eliminando el estudiante"
Example:

```
URL: {server url}/api/students/2088654382
Body: { }
Returns
{
   "message": "Se eliminó el estudiante exitosamente"
}
```

Teachers

Create Teacher

Description: Create student
Method: POST
Route: /teachers
Errors: "Falta un campo del profesor", "Carnet ya está registrado", "Correo ya está registrado", "Grado inválido", "Un error ocurrió enviando el correo con la contraseña", "Un error ocurrió creando el profesor"
Example:

```
URL: {server url}/api/teachers
Body: {
   "id": "2088654382",
   "email": "armando@gmail.com",
   "name": "Armando",
   "lastname": "Arias Chacon"
}
Returns
{
   "message": "Se creó el profesor exitosamente"
}
```

Get Teacher

Description: Get teacher by id. Ratings come with embedded student documents.
Method: GET
Route: /teachers/id
Errors: "Profesor no existe", "Un error ocurrió recuperando el profesor"
Example:

```
URL: {server url}/api/teachers/706483322
Body: { }
Returns
{
   "teacher": {
      "_id": "6115495572a28106b8ec56d0",
      "id": "706483322",
      "email": "mario@gmail.com",
      "password": "$2a$10$NQBma9qcIS4Oc9r9U6Fu0ea1TG3X5G/r3SnlQKz.hAcuShfR07S9a",
      "name": "Mario",
      "lastname": "Yoses Vindas",
      "ratings": [
         {
            "_id": "61156fef12d0e02470dcdc99",
            "student": {
                "_id": "611369eba56e672ba8fe781b",
                "id": "206754833",
                "email": "juan@gmail.com",
                "password": "$2a$10$uM7P5ynobi7Bwd1NHVsv1ehr46Mn9z6j7eEfilYck5ZQN5Jol.2.C",
                "name": "Juan",
                "lastname": "Rojas Vargas",
                "grade": 6,
                "__v": 0
            },
            "rating": 100
        }
    ],
    "__v": 7
  }
}
```

Get All Teachers

Description: Get all teachers
Method: GET
Route: /teachers
Errors: "Un error ocurrió recuperando todos los profesores"
Example:

```
URL: {server url}/api/teachers
Body: { }
Returns
{
  "teachers": [
    {
      "_id": "611546da399b5849ec69b2cb",
      "id": "206483322",
      "email": "gary@gmail.com",
      "password": "$2a$10$z9iQ7qndTh08/EiGtu6ZHeAJU.j4niRxHeftaTMXvhrlWlLSYIqBy",
      "name": "Gary",
      "lastname": "Heinz Baldez",
      "ratings": [],
      "__v": 0
    },
    {
      "_id": "6115495572a28106b8ec56d0",
      "id": "706483322",
      "email": "mario@gmail.com",
      "password": "$2a$10$NQBma9qcIS4Oc9r9U6Fu0ea1TG3X5G/r3SnlQKz.hAcuShfR07S9a",
            "name": "Mario",
            "lastname": "Yoses Vindas",
            "ratings": [
              {
                "_id": "61156fef12d0e02470dcdc99",
                "student": {
                  "_id": "611369eba56e672ba8fe781b",
                  "id": "206754833",
                  "email": "juan@gmail.com",
                  "password": "$2a$10$uM7P5ynobi7Bwd1NHVsv1ehr46Mn9z6j7eEfilYck5ZQN5Jol.2.C",
                  "name": "Juan",
                  "lastname": "Rojas Vargas",
                  "grade": 6,
                  "__v": 0
                },
                "rating": 100
              }
            ],
            "__v": 7
          }
        ]
      }
```

Update Teacher

Description: Update teacher by id.
Method: PUT
Route: /teachers/id
Errors: "Falta un campo del profesor", "Correo ya está en uso", "Profesor no existe", "Un error ocurrió actualizando el profesor"
Example:

URL: {server url}/api/teachers/706483322

Body:

```
{
  "email": "mario@gmail.com",
  "name": "Mario",
  "lastname": "Yoses Meses"
}
```

Returns

```
{
  "message": "Se actualizó el profesor exitosamente"
}
```

Delete Teacher

Description: Delete teacher by id
Method: DELETE
Route: /teachers/id
Errors: "Profesor no existe", "No se puede eliminar un profesor con cursos asignados", "Un error ocurrió eliminando el profesor"
Example:

```
URL: http://localhost:3000/api/teachers/706483322
Body: { }
Returns
{
    "message": "Se eliminó el profesor exitosamente"
}
```

Teacher Ratings

Create/Update Teacher Rating

Description: Creates a teacher rating. If the student already rated the teacher the rating number is just updated.
Method: PUT
Route: /teachers/id/ratings
Errors: "Estudiante no existe", "Profesor no existe", "Calificación inválida (debe ser de 0 a 100)!", "Un error ocurrió calificando el profesor"
Example:

```
URL: {server url}/api/teachers/706483322/ratings
Body:
{
   "studentId": "206754833",
   "rating": 5
}
Returns
{
   "message": "Calificación de profesor exitosa"
}
```

Delete Teacher Rating

Description: Deletes a teacher rating by student id.
Method: DELETE
Route: /teachers/teacherId/ratings/studentId
Errors: "Estudiante no existe", "Profesor no existe", "Un error ocurrió eliminando la calificación del profesor"
Example:

```
URL: {server url}/api/teachers/706483322/ratings/206754833
Body: {}
Returns
{
   "message": "Eliminación de calificación de profesor exitosa"
}
```

Courses

Create Course

Description: Create course
Method: POST
Route: /courses
Errors: "Falta un campo del curso", "Grado inválido", “Id de curso ya está registrado”, "Un error ocurrió creando el curso"
Example:

```
URL: {server url}/api/courses
Body:
{
   "id": "mat01",
  "name": "Matematicas 1",
   "grade": 1
}
Returns
{
   "message": "Se creó el curso exitosamente"
}
```

Get Course

Description: Get course by id
Method: GET
Route: /courses/id
Errors: "Curso no existe", "Un error ocurrió recuperando el curso"
Example:

```
URL: {server url}/api/courses/math01
Body: { }
Returns
{
   "course": {
      "students": [],
      "_id": "61280bf34dfe460e787c77a7",
      "id": "mat01",
      "name": "Matematicas 1",
      "grade": 1,
      "news": [],
      "schedule": [
         {
            "_id": "612812b5dca5ab400c42556c",
            "day": 1,
            "startHour": 7,
                "startMinutes": 30,
                "endHour": 9,
                "endMinutes": 20
            }
        ],
        "assignments": [],
        "chat": [],
        "__v": 1,        "__v": 0
        "teacher": {
            "_id": "611546da399b5849ec69b2cb",
            "id": "206483322",
            "email": "gary@gmail.com",
            "password": "$2a$10$z9iQ7qndTh08/EiGtu6ZHeAJU.j4niRxHeftaTMXvhrlWlLSYIqBy",
            "name": "Gary",
            "lastname": "Heinz Baldez",
            "ratings": [
                {
                    "_id": "612865bb3f19dc00169339bc",
                    "student": {
                        "_id": "611369eba56e672ba8fe781b",
                        "id": "206754833",
                        "email": "juan@gmail.com",
                        "password": "$2a$10$uM7P5ynobi7Bwd1NHVsv1ehr46Mn9z6j7eEfilYck5ZQN5Jol.2.C",
                        "name": "Juan",
                        "lastname": "Rojas Vargas",
                        "grade": 6,
                        "__v": 0
                    },
                    "rating": 5
                }
            ],
            "__v": 1
        }
    }
}
```

Get Courses

Description: Get courses, with optionals teacher (studentId) and student (studentId) filters as query parameters. Only one is applied and the teacher is prioritized.

Method: GET

Route: /courses

Errors: "Profesor no existe", "Un error ocurrió recuperando los cursos"

Example:

URL: {server url}/api/courses?teacherId=206483322
Body: { }
Returns

```
{
  "courses": [
    {
      "students": [],
      "_id": "61280bfc4dfe460e787c77ac",
      "id": "mat02",
      "name": "Matematicas 2",
      "grade": 2,
      "news": [],
      "schedule": [
        {
          "_id": "612812d0dca5ab400c425571",
          "day": 1,
          "startHour": 8,
          "startMinutes": 0,
          "endHour": 10,
          "endMinutes": 0
        }
      ],
      "assignments": [],
      "chat": [],
      "__v": 1
    }
  ]
}
```

Update Course

Description: Update course basic information by id
Method: PUT
Route: /courses/id
Errors: "Falta un campo del curso", "Grado inválido", “Curso no existe”, "Un error ocurrió actualizando el curso"
Example:

URL: {server url}/api/courses/mat01
Body:

```
{
  "name": "Matematicas 1",
  "grade": 3
}
```

```
Returns
{
    "message": "Se actualizó el curso exitosamente"
}
```

Delete Course

Description: Delete course by id
Method: DELETE
Route: /courses/id
Errors: "Curso no existe", "Un error ocurrió eliminando el curso"
Example:

```
URL: http://localhost:3000/api/courses/esp01
Body: { }
Returns
{
    "message": "Se eliminó el curso exitosamente"
}
```

Course Schedule

Create Schedule Class Period

Description: Add a class period to a course
Method: POST
Route: /courses/id/schedule
Errors: "Falta un campo del periodo de clase", “Curso no existe”, "Número de día inválido", “Hora inicial o final inválida”, “Minutos iniciales o finales inválidos”, "Nuevo periodo de clase choca con el horario del curso", "Horario de curso no puede ser modificado mientras haya estudiantes inscritos y un profesor asignado", "Un error ocurrió añadiendo el periodo de clase del curso"
Example:

```
URL: {server url}/api/courses/mat01/schedule
Body:
{
   "day": 1,
   "startHour": 7,
   "startMinutes": 30,
   "endHour": 9,
   "endMinutes": 0
}
Returns
{
   "message": "El nuevo periodo de clase se añadió exitosamente"
}
```

Delete Schedule Class Period

Description: Delete a class period to a course using mongodb id
Method: DELETE
Route: /courseId/schedule/classPeriodId
Errors: "Falta el id del periodo de clase", “Curso no existe”, “Período de clase no existe”, "Horario de curso no puede ser modificado mientras haya estudiantes inscritos y un profesor asignado", "Un error ocurrió eliminando el periodo de clase"
Example:

```
URL: {server url}/api/courses/mat01/schedule/6127f74ef096482be88785de
Body: { }
Returns
{
   "message": "Eliminación de periodo de clase exitosa"
}
```

Course Teacher

Update Course Teacher

Description: Updates the course teacher, overrides current one. Setting the same teacher multiple times doesn't generate errors. If no teacher is assigned the course, the document gains the teacher attribute. Is checked that the course schedule doesn't overlap the schedule of other courses assigned to the teacher.
Method: PUT
Route: /courses/id/teacher
Errors: "Falta el id del profesor", "Curso no existe", "Profesor no existe", "Horario de clases del profesor choca con horario de curso", "Un error ocurrió actualizando el profesor del curso"
Example:

```
URL: {server url}/api/courses/mat01/teacher
Body:
{
  "teacherId": "206483322"
}

Returns
{
  "message": "Se actualizó el profesor del curso exitosamente"
}
```

Delete Course Teacher

Description: Deletes the teacher attribute from the course document.
Method: DELETE
Route: /courses/id/teacher
Errors: "Curso no existe", "No se puede eliminar el profesor de un curso con estudiantes inscritos", "Un error ocurrió removiendo el profesor del curso"
Example:

```
URL: {server url}/api/courses/mat01/teacher
Body:
{
  "teacherId": "206483322"
}

Returns
{
  "message": "Se removió el profesor del curso exitosamente"
}
```

Course Students

Create Course Student

Description: Enrolls a student on a course, checks for schedule overlapping.
Method: POST
Route: /courses/id/students
Errors: "Falta el id del estudiante", “Curso no existe”, “Estudiante no existe”, “Grado de estudiante y curso no coinciden”, “Curso no tienen un horario definido todavía”, “Curso no tiene un profesor asignado todavía”, “Estudiante ya está inscrito en el curso”, “Horario del estudiante choca con horario del curso ”, "Un error ocurrió inscribiendo el estudiante"
Example:

```
URL: {server url}/api/courses/mat01/students
Body:
{
   "studentId": "205746383"
}
Returns
{
   "message": "Se inscribió el estudiante exitosamente"
}
```

Delete Course Student

Description: Disenrolls a student from a course
Method: DELETE
Route: /courses/courseId/students/studentId
Errors: "Falta el id del estudiante", “Curso no existe”, “Estudiante no existe”, “Estudiante no está inscrito en el curso”, "Un error ocurrió dando de baja al estudiante"
Example:

```
URL: {server url}/api/courses/mat01/students/205746383
Body: { }
Returns
{
   "message": "Se dio de baja al estudiante exitosamente"
}
```

Email Students List

Description: Emails a course’s list of students as a pdf table.
Method: POST
Route: /courses/id/students/email-list
Errors: "Falta el correo electrónico", “Curso no existe”, "Un error ocurrió enviando la lista de estudiantes", "No se pudo enviar el correo con los estudiantes del curso"
Example:

URL: {server url}/api/courses/mat01/students/email-list
Body:

```
{
    "email": "mepdigitalweb.prof1@gmail.com"
}
```

Returns

```
{
    "message": "Se envió exitosamente la lista de estudiantes"
}
```

Course News

Create Course News

Description: Create course news
Method: POST
Route: /courses/id/news
Errors: "Falta un campo de la noticia de curso", "Curso no existe", "Un error ocurrió creando la noticia de curso"
Example:

```
URL: {server url}/api/courses/mat01/news
Body:
{
   "title": "Primer examen",
   "message": "Este primer examen es de sumas y restas",
   "date": "2021-08-31T02:54:11.722Z"
}
Returns
{
   "message": "Se creó la noticia del curso exitosamente"
}
```

Update Course News

Description: Update course news by id
Method: PUT
Route: /courses/courseId/news/newsId
Errors: "Falta un campo de la noticia de curso", "Curso no existe", “Noticia de clase no existe”, "Un error ocurrió actualizando la noticia de curso"
Example:

```
URL: {server url}/api/courses/mat01/news/612d9a7f998ab646c4546c89
Body:
{
   "title": "Primer examen",
   "message": "Este primer examen es de sumas y restas"
}
Returns
{
   "message": ”Actualización de noticia de curso exitosa"
}
```

Delete Course News

Description: Delete course news by id

Method: DELETE
Route: /courses/courseId/news/newsId
Errors: "Curso no existe", “Noticia no existe”, “Noticia de clase no existe”, "Un error ocurrió eliminando la noticia de curso"
Example:

```
URL: {server url}/api/courses/mat01/news/612d9a7f998ab646c4546c89
Body: { }
Returns
{
    "message": ”Eliminación de noticia de curso exitosa"
}
```

Course Assignments

Create Course Assignment

Description: Create course assignment
Method: POST
Route: /courses/id/assignment
Errors: "Falta un campo de la tarea de curso", "Curso no existe", “Tarea de curso no existe”, "Un error ocurrió creando la tarea de curso"
Example:

```
URL: {server url}/api/courses/mat01/assignments
Body:
{
   "title": "Tarea 1",
   "description": "Sumar 3+5",
   "submitDate": "2021-08-31T02:54:11.722Z"
}
Returns
{
   "message": "Se creó la tarea del curso exitosamente"
}
```

Update Course Assignment

Description: Update course assignment by id
Method: PUT
Route: /courses/courseId/assignment/assignmentId
Errors: "Falta un campo de la tarea de curso", "Curso no existe", “Tarea de clase no existe”, "Un error ocurrió actualizando la noticia de curso"
Example:

```
URL: {server url}/api/courses/mat01/assignments/612d9fb67da2724c382953fa
Body:
{
   "title": "Tarea 1",
   "description": "Sumar 3+9",
   "submitDate": "2021-08-30T02:54:11.722Z"
}
Returns
{
   "message": ”Actualización de tarea de curso exitosa"
}
```

Delete Course Assignment

Description: Delete course assignment by id

Method: DELETE
Route: /courses/courseId/assignment/assignmentId
Errors: "Curso no existe", “Tarea de curso no existe”, "Un error ocurrió eliminando la tarea de curso"
Example:

```
URL: {server url}/api/courses/mat01/assignments/612d9fb67da2724c382953fa
Body: { }
Returns
{
   "message": ”Eliminación de tarea de curso exitosa"
}
```

Course Chat

Create Course Chat Message

Description: Create course chat message. UserType must be ‘Teacher’ or ‘Student’.
Method: POST
Route: /courses/id/chat
Errors: "Falta un campo del mensaje de chat", "Curso no existe", “Tipo de usuario inválido, debe ser estudiante o profesor”, “Usuario profesor no es el profesor asignado del curso”, “Usuario estudiante no está inscrito en el curso”, "Un error ocurrió creando el mensaje de chat"
Example:

```
URL: {server url}/api/courses/mat01/chat
Body:
{
   "userId": "205746383",
   "userType": "Student",
   "message": "¡Hola profe!"
}
Returns
{
   "message": "Se creó el mensaje de chat exitosamente"
}
```

Update Course Chat Message

Description: Update course chat message by id
Method: PUT
Route: /courses/courseId/chat/chatMessageId
Errors: "Falta un campo del mensaje de chat", "Curso no existe", “Mensaje de chat no existe”, "Un error ocurrió creando el mensaje de chat"
Example:

```
URL: {server url}/api/courses/mat01/chat/612dc97e887a513dd0da89b5
Body:
{
   "message": "Hola profe"
}
Returns
{
   "message": ”Actualización de mensaje de chat exitosa"
}
```

Delete Course Chat Message

Description: Delete course chat message by id
Method: DELETE

Route: /courses/courseId/chat/chatMessageId
Errors: "Falta un campo del mensaje de chat", "Curso no existe", “Mensaje de chat no existe”, "Un error ocurrió eliminando el mensaje de chat"
Example:

```
URL: {server url}/api/courses/mat01/chat/612dc97e887a513dd0da89b5
Body: { }
Returns
{
   "message": ”Eliminación de mensaje de chat exitosa"
}
```