CURSOS BOOTCAMP

INSTRUCCIONES

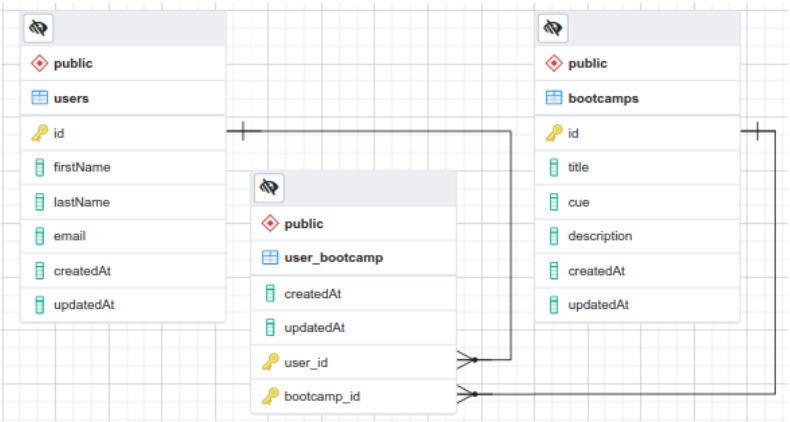
Lee con atención cada uno de los requerimientos que se presentan a continuación, y desarrolla la prueba de acuerdo con lo solicitado.

DESCRIPCIÓN

Planteamiento del ejercicio de aplicación.

Actualmente, el equipo de desarrollo de software emite un requerimiento, donde se desea diseñar la gestión de cursos Bootcamp de una determinada empresa de adiestramiento. El equipo aplica la Metodología Scrum, y realiza el primer Sprint que trata de elaborar el proyecto por medio de Node.js, el registro de cursos Bootcamp y de usuarios de éstos.

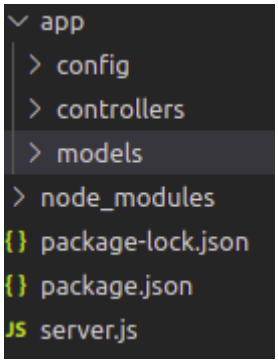
En este primer Sprint se obtiene el diseño inicial de la base de datos, el cual consta del registro de usuarios y de cursos Bootcamp, obteniendo el siguiente Modelo de Entidad - Relación de la base de datos:



El requerimiento emitido por la empresa de adiestramiento parte del principio de que los usuarios pueden participar en distintos Bootcamp, y, a su vez, distintos Bootcamp poseen distintos usuarios.

Para esta primera parte del Sprint, el equipo de desarrollo, junto con el ScrumMaster, acordaron los siguientes pasos a seguir:

1. Hacer uso de Node.JS para el desarrollo.
2. Partir de la siguiente estructura para el desarrollo:



3. Crear dentro de la carpeta **config**, el archivo **db.config.js**, que posee la configuración a la base de datos; el nombre de ésta es: **db_bootcamp.**
4. Dentro de la carpeta **models**, se encuentran los modelos tanto para el usuario (**user.model.js**), como para el Bootcamp (**bootcamp.model.js**)

   Los atributos para el usuario son:

   - **firstName**: cadena de caracteres y campo obligatorio.
   - **lastName**: cadena de caracteres y campo obligatorio.
   - **email**: campo obligatorio, y con las siguientes validaciones: formato de correo y que sea único, no repetitivo en la base de datos.

   Los atributos para el Bootcamp son:

   - **title**: cadena de caracteres que define el título del Bootcamp, campo obligatorio.
   - **cue**: número que define la cantidad de sesiones (CUE) que contiene el Bootcamp, campo obligatorio con las siguientes validaciones: tipo entero con un valor mínimo de 5 CUE y como máximo 10.
   - **description**: cadena de caracteres que define la descripción o el objetivo del Bootcamp, campo obligatorio.

   Y el archivo i**ndex.js**, donde se define la conexión con sequelize a la base de datos y modelos.

5. En la carpeta **controllers** posee los controladores tanto para el usuario (**user.controller.js**), como para el Bootcamp (**bootcamp.controller.js**).

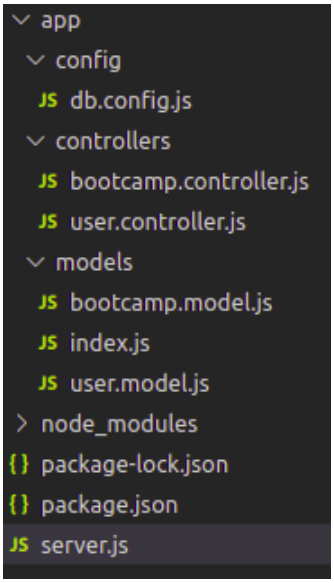
   Para el usuario, construir los siguientes controladores:

   - Crear y guardar usuarios llamado **createUser.**
   - Obtener los Bootcamp de un usuario llamado **findUserById.**
   - Obtener todos los Usuarios incluyendo, los Bootcamp llamado **findAll.**
   - Actualizar usuario por Id llamado **updateUserById.**
   - Eliminar un usuario por Id llamado **deleteUserById.**

   Para el Bootcamp, construir los siguientes controladores:

   - Crear y guardar un nuevo Bootcamp llamado **createBootcamp.**
   - Agregar un Usuario al Bootcamp llamado **addUser.**
   - Obtener los Bootcamp por id llamado **findById**.
   - Obtener todos los Usuarios incluyendo los Bootcamp llamado **findAll.**

   La estructura final es:



6. Por último, para verificar los modelos y las relaciones con sus métodos, se crea el archivo **server.js**, donde hacemos uso de la base de datos, los modelos y los controladores.

   Crear los siguientes usuarios:

firstName	lastName	email
Mateo	Díaz	mateo.diaz@correo.com
Santiago	Mejías	santiago.mejias@correo.com
Lucas	Rojas	lucas.rojas@correo.com
Facundo	Fernandez	facundo.fernandez@correo.com

   Al ejecutar la creación de usuario, tenemos en la terminal la siguiente salida:

```
  node server.js

Executing (default): DROP TABLE IF EXISTS "user_bootcamp" CASCADE;
Executing (default): DROP TABLE IF EXISTS "bootcamps" CASCADE;
Executing (default): DROP TABLE IF EXISTS "users" CASCADE;
Executing (default): DROP TABLE IF EXISTS "users" CASCADE;
Executing (default): CREATE TABLE IF NOT EXISTS "users" ("id"   SERIAL , "firstName" VARCHAR(255), "lastName"
VARCHAR(255), "email" VARCHAR(255), "createdAt" TIMESTAMP WITH TIME ZONE NOT NULL, "updatedAt" TIMESTAMP
WITH TIME ZONE NOT NULL, UNIQUE ("email"), PRIMARY KEY ("id"));
Executing (default): SELECT i.relname AS name, ix.indisprimary AS primary, ix.indisunique AS unique, ix.indkey AS indkey,
array_agg(a.attnum) as column_indexes, array_agg(a.attname) AS column_names, pg_get_indexdef(ix.indexrelid) AS definition FROM
pg_class t, pg_class i, pg_index ix, pg_attribute a WHERE t.oid = ix.indrelid AND i.oid = ix.indexrelid AND a.attrelid = t.oid AND
t.relkind = 'r' and t.relname = 'users' GROUP BY i.relname, ix.indexrelid, ix.indisprimary, ix.indisunique, ix.indkey ORDER BY i.relname;
Executing (default): DROP TABLE IF EXISTS "bootcamps" CASCADE;
Executing (default): CREATE TABLE IF NOT EXISTS "bootcamps" ("id"   SERIAL , "title" VARCHAR(255), "cue" INTEGER,
"description" VARCHAR(255), "createdAt" TIMESTAMP WITH TIME ZONE NOT NULL, "updatedAt" TIMESTAMP WITH TIME
ZONE NOT NULL, PRIMARY KEY ("id"));
Executing (default): SELECT i.relname AS name, ix.indisprimary AS primary, ix.indisunique AS unique, ix.indkey AS indkey,
array_agg(a.attnum) as column_indexes, array_agg(a.attname) AS column_names, pg_get_indexdef(ix.indexrelid) AS definition FROM
pg_class t, pg_class i, pg_index ix, pg_attribute a WHERE t.oid = ix.indrelid AND i.oid = ix.indexrelid AND a.attrelid = t.oid AND
t.relkind = 'r' and t.relname = 'bootcamps' GROUP BY i.relname, ix.indexrelid, ix.indisprimary, ix.indisunique, ix.indkey ORDER BY
i.relname;
Executing (default): DROP TABLE IF EXISTS "user_bootcamp" CASCADE;
Executing (default): CREATE TABLE IF NOT EXISTS "user_bootcamp" ("createdAt" TIMESTAMP WITH TIME ZONE NOT NULL,
"updatedAt" TIMESTAMP WITH TIME ZONE NOT NULL, "user_id" INTEGER  REFERENCES "users" ("id") ON DELETE
CASCADE ON UPDATE CASCADE, "bootcamp_id" INTEGER  REFERENCES "bootcamps" ("id") ON DELETE CASCADE ON
UPDATE CASCADE, PRIMARY KEY ("user_id","bootcamp_id"));
Executing (default): SELECT i.relname AS name, ix.indisprimary AS primary, ix.indisunique AS unique, ix.indkey AS indkey,
array_agg(a.attnum) as column_indexes, array_agg(a.attname) AS column_names, pg_get_indexdef(ix.indexrelid) AS definition FROM
pg_class t, pg_class i, pg_index ix, pg_attribute a WHERE t.oid = ix.indrelid AND i.oid = ix.indexrelid AND a.attrelid = t.oid AND
t.relkind = 'r' and t.relname = 'user_bootcamp' GROUP BY i.relname, ix.indexrelid, ix.indisprimary, ix.indisunique, ix.indkey ORDER BY
i.relname;
Eliminando y resincronizando la base de datos.
Executing (default): INSERT INTO "users" ("id","firstName","lastName","email","createdAt","updatedAt") VALUES
(DEFAULT,$1,$2,$3,$4,$5) RETURNING "id","firstName","lastName","email","createdAt","updatedAt";
>> Se ha creado el usuario: {
   "id": 1,
   "firstName": "Mateo",
   "lastName": "Díaz",
   "email": "mateo.diaz@correo.com",
   "updatedAt": "2022-04-23T18:06:22.251Z",
   "createdAt": "2022-04-23T18:06:22.251Z"
}
```

```
Executing (default): INSERT INTO "users" ("id","firstName","lastName","email","createdAt","updatedAt") VALUES
(DEFAULT,$1,$2,$3,$4,$5) RETURNING "id","firstName","lastName","email","createdAt","updatedAt";
>> Se ha creado el usuario: {
  "id": 2,
  "firstName": "Santiago",
  "lastName": "Mejias",
  "email": "santiago.mejias@correo.com",
  "updatedAt": "2022-04-23T18:06:22.307Z",
  "createdAt": "2022-04-23T18:06:22.307Z"
}
Executing (default): INSERT INTO "users" ("id","firstName","lastName","email","createdAt","updatedAt") VALUES
(DEFAULT,$1,$2,$3,$4,$5) RETURNING "id","firstName","lastName","email","createdAt","updatedAt";
>> Se ha creado el usuario: {
  "id": 3,
  "firstName": "Lucas",
  "lastName": "Rojas",
  "email": "lucas.rojas@correo.com",
  "updatedAt": "2022-04-23T18:06:22.318Z",
  "createdAt": "2022-04-23T18:06:22.318Z"
}
Executing (default): INSERT INTO "users" ("id","firstName","lastName","email","createdAt","updatedAt") VALUES
(DEFAULT,$1,$2,$3,$4,$5) RETURNING "id","firstName","lastName","email","createdAt","updatedAt";
>> Se ha creado el usuario: {
  "id": 4,
  "firstName": "Facundo",
  "lastName": "Fernández",
  "email": "facundo.fernandez@correo.com",
  "updatedAt": "2022-04-23T18:06:22.330Z",
  "createdAt": "2022-04-23T18:06:22
```

Crear los siguientes Bootcamp:

title	cue	description
Introduciendo El Bootcamp De React.	10	React es la librería más usada en JavaScript para el desarrollo de interfaces.
Bootcamp Desarrollo Web Full Stack.	12	Crearás aplicaciones web utilizando las tecnologías y lenguajes más actuales y populares, como: JavaScript, nodeJS, Angular, MongoDB, ExpressJS.
Bootcamp Big Data, Inteligencia Artificial & Machine Learning.	18	Domina Data Science, y todo el ecosistema de lenguajes y herramientas de Big Data, e intégralos con modelos avanzados

		de Artificial Intelligence y Machine Learning.

Al ejecutar la creación de los Bootcamp, se tiene en la salida del terminal lo siguiente:

```
$ node server.js

Executing (default): INSERT INTO "bootcamps" ("id","title","cue","description","createdAt","updatedAt") VALUES
(DEFAULT,$1,$2,$3,$4,$5) RETURNING "id","title","cue","description","createdAt","updatedAt";
>> Creado el bootcamp: {
  "id": 1,
  "title": "Introduciendo El Bootcamp De React",
  "cue": 10,
  "description": "React es la librería más usada en JavaScript para el desarrollo de interfaces",
  "updatedAt": "2022-04-23T18:14:06.694Z",
  "createdAt": "2022-04-23T18:14:06.694Z"
}
Executing (default): INSERT INTO "bootcamps" ("id","title","cue","description","createdAt","updatedAt") VALUES
(DEFAULT,$1,$2,$3,$4,$5) RETURNING "id","title","cue","description","createdAt","updatedAt";
>> Creado el bootcamp: {
  "id": 2,
  "title": "Bootcamp Desarrollo Web Full Stack",
  "cue": 12,
  "description": "Crearás aplicaciones web utilizando las tecnologías y lenguajes más actuales y populares como JavaScript, nodeJS,
Angular, MongoDB, ExpressJS",
  "updatedAt": "2022-04-23T18:14:06.704Z",
  "createdAt": "2022-04-23T18:14:06.704Z"
}
Executing (default): INSERT INTO "bootcamps" ("id","title","cue","description","createdAt","updatedAt") VALUES
(DEFAULT,$1,$2,$3,$4,$5) RETURNING "id","title","cue","description","createdAt","updatedAt";
>> Creado el bootcamp: {
  "id": 3,
  "title": "Bootcamp Big Data, Inteligencia Artificial & Machine Learning",
  "cue": 12,
  "description": "Domina Data Science todo el ecosistema de lenguajes y herramientas de Big Data e intégralos con modelos avanzados de
Artificial Intelligence y Machine Learning",
  "updatedAt": "2022-04-23T18:14:06.714Z",
  "createdAt": "2022-04-23T18:14:06.714Z"
```

Agregar los siguientes usuarios al Bootcamp:

title	usuarios
Introduciendo El Bootcamp De React.	Mateo Díaz Santiago Mejías
Bootcamp Desarrollo Web Full Stack.	Mateo Díaz

Bootcamp Big Data, Inteligencia Artificial & Machine Learning.	Mateo Díaz Santiago Mejías Lucas Rojas

La salida en la terminal es:

```
$ node server.js

****************************
 Agregado el usuario id=1 al bootcamp con id=1
****************************

****************************
 Agregado el usuario id=2 al bootcamp con id=1
****************************

****************************
 Agregado el usuario id=1 al bootcamp con id=2
****************************

****************************
 Agregado el usuario id=1 al bootcamp con id=3
****************************

****************************
 Agregado el usuario id=2 al bootcamp con id=3
****************************

****************************
 Agregado el usuario id=3 al bootcamp con id=3
```

Realizar las siguientes consultas:

- Consultando el Bootcamp por id, incluyendo los usuarios.
- Listar todos los Bootcamp con sus usuarios.
- Consultar un usuario por id, incluyendo los Bootcamp.
- Listar los usuarios con sus Bootcamp.
- Actualizar el usuario según su id; por ejemplo: actualizar el usuario con id=1 por Pedro Sánchez.
- Eliminar un usuario por id; por ejemplo: el usuario con id=1.