

CE 5508

Parking-TEC

INVESTIGACION #1

Presentado por:
Esteban Alvarado
Martín Calderón
Olman Castro

HTTPS

Este protocolo junta a HTTP y le agrega seguridad utilizando los certificados SSL/TLS.

¡Vamos a implementarlo en nuestra API de Parking TEC!



Se implementa una conexión https al API en el puerto 8080 y se mantiene el puerto 8000 para conexión http.

```
[nodemon] 2.0.12
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index index.js`
Server running 🚀 on port: 8000...
Secure server 🚀🔑 on port: 8080...
█
```

HTTP

: 8000

HTTPS

: 8080

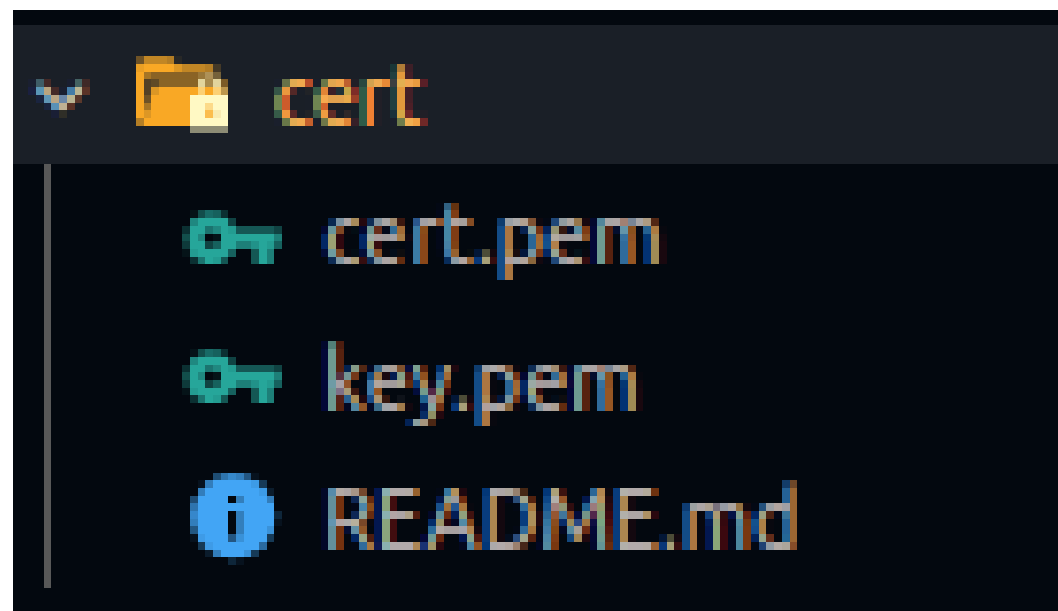


Certificado SSL

SE generaron las llaves y el certificado Self-Signed utilizando **openssl** y el modulo https de node js.

/cert

Almacena los archivos de la llave privada y el certificado.



key



```
-----BEGIN PRIVATE KEY-----
MIIEQgIBADANBgkqhkiG9w0BAQEFAASCCSwggkoAgEAAoICAQDHgynGylxCnqVS
Y55/+KaeMVJAmLGCCEcBBaCmf+R4NQ7GKf/tB2IlrhIAk49opwytSUeLU0xREuS
9+9zLpeJ6aa1xs8LsgQcZ668uHv7zsWXJl6D5CERYW8dy7/eZhnURuvhl6pOP8P8
BGBGxP9HdaIYbqWR2N04CmaCzs4Nnd3DxyBnht+NiWfBComfmEtXfzJ5Vjo14S8
j0xcbbSHiR/hAjSt+jj3Nkg6PL3HHKD/w69v0rXqkMNlyo9YqxFSVUIWFrUxxZms
rPPyYrKfysy+LPI6pTQsDZzTOE7UoWhPSAn7QLkTo0oLwoFhPAYhCVJjflc9bpvK
ZzwDa9jcyBaNDuZIETo88kEgBbzBfHqu5M+EnHKTxutv6vwz7YfpuUeUroqQwRle
Qj+Rjql0l04DMLGQVeAfTNAqZ2iF7dFllMEVKLn timer6upyJwbz1UL5qJwknlp
irrKyXxMZGEBLHETDD5xpURLVTjxybryFn8GX8VVYNPs065CKWk5DFdgkfBxSjJr
c2hpHtshnktJX5LL1PHvSPNeotND1fG9lh9qwtKf+M6Lw9vteHvcC+L9AJi/zh+p
llzWpBLL+Fy2+JXQZohUpyU8KnF3ZPk1f5XdNb+exE51Nc7UTdpjKBjqapSfgiU2
Z170uZ6LzqVQhEabAB4QphTcu7fWEQIDA0ABAoICAQCMv6szfJ5h08yu8oXiKvjx
K/697moIce8aLZ/XUvio3oK5f0dvHEsbgu+vQy4TqgTZesKx5wv4eUZNWmaffwZF
0EcFlW4nMu5o602W7Q0KwbwrRfKihSh4JWMGH/8KaYWSiLBP a8J/pK0un+HliGB7
V4bqsiZNpmZboaZ0GaL+k0qBkIowwvd2q8IMDSLIAURmcLsT91+7JGFewNqJ81WI
MkWqkX0FN7o0wZzkxJswCYM4EFst05XGwrv4772jvXC/ynNzkAxvzeLKCVXxAD6h
hKRjeJaCHPdffQpVM7AD5+US0wE1Qsj39/3W+LFNnybom6h7cXZx73TjZCcyHQmx
IMMiNpjU00Zb00ggtS2C5kjiLqOXQrDlHnQ28qD89kZlifoLPDp0s/De4qLJv96D
ldkfYs0AtncySiCsgqWIXJpexJHmvUwlo3kH23+P8gzQf1rwDup9Ymd173BN0hfS
KLG3Cx8XvvbbIKKjKDsPBv94vVaQoeM6aXguSpcbbWxk6G33samI7kMvGklJhziq
Y7zflswDYk1JK08/bRt rJNqHW1cEgwJ7/4v0/t06pnt2KHky836AVNxGpzrQDwC
Bw5XbzYjA0lCgF0Armd i fbo2aQNEGb7vNBZcJ/QQd0w/ABnK2QvwBFSRaFgJ1+63
h2Z2WmNGiSUDciFcQzei0QKCAQEA5z6vHTQuf7Ts q5PYEcjx74c4z+C/okmwixkc
d54MXTBUmT33j38LC5lt5r6JHgVAcCA2XXY4u7+vjqF0ybPntPOS3FCQq9fSgDkp
14KuoTrXigeDf3kiK0Dez1b1d6Q3WAlb6bFolUwku8+tlUvzPFAcgaMiguyMuwDk4b
```

La llave privada generada.



Certificado SSL

SE generaron las llaves y el certificado Self-Signed utilizando **openssl** y el modulo https de node js.

index.js

Se inicia el servidor con https

```
const app = express();
app.use(express.json());
app.use(cors());

const sslServer = https.createServer(config.ssl, app);
```

¡Self-Signed Certificate!

config

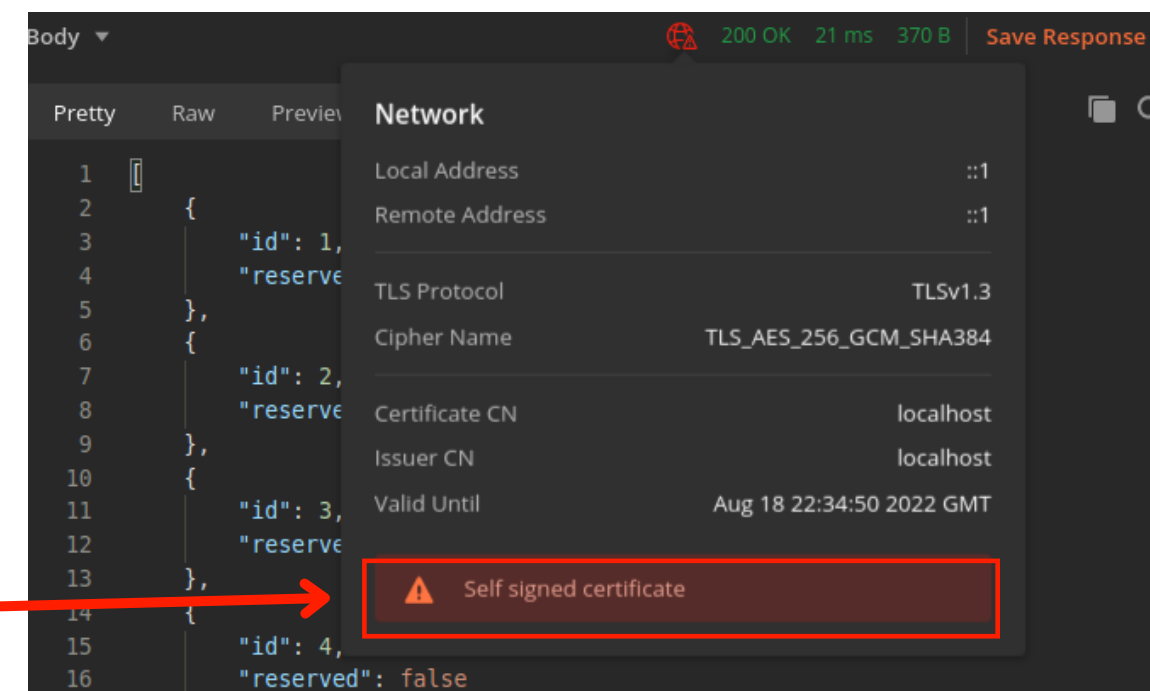
En la configuración se crea una propiedad ssl

```
ssl: {
  key: fs.readFileSync(path.join(__dirname, '../cert/key.pem')),
  cert: fs.readFileSync(path.join(__dirname, '../cert/cert.pem'))
}
```

postman

¡Ahora se puede llamar con https!

https://localhost:8080/api/spaces?filter=id,reserved







Certificado SSL

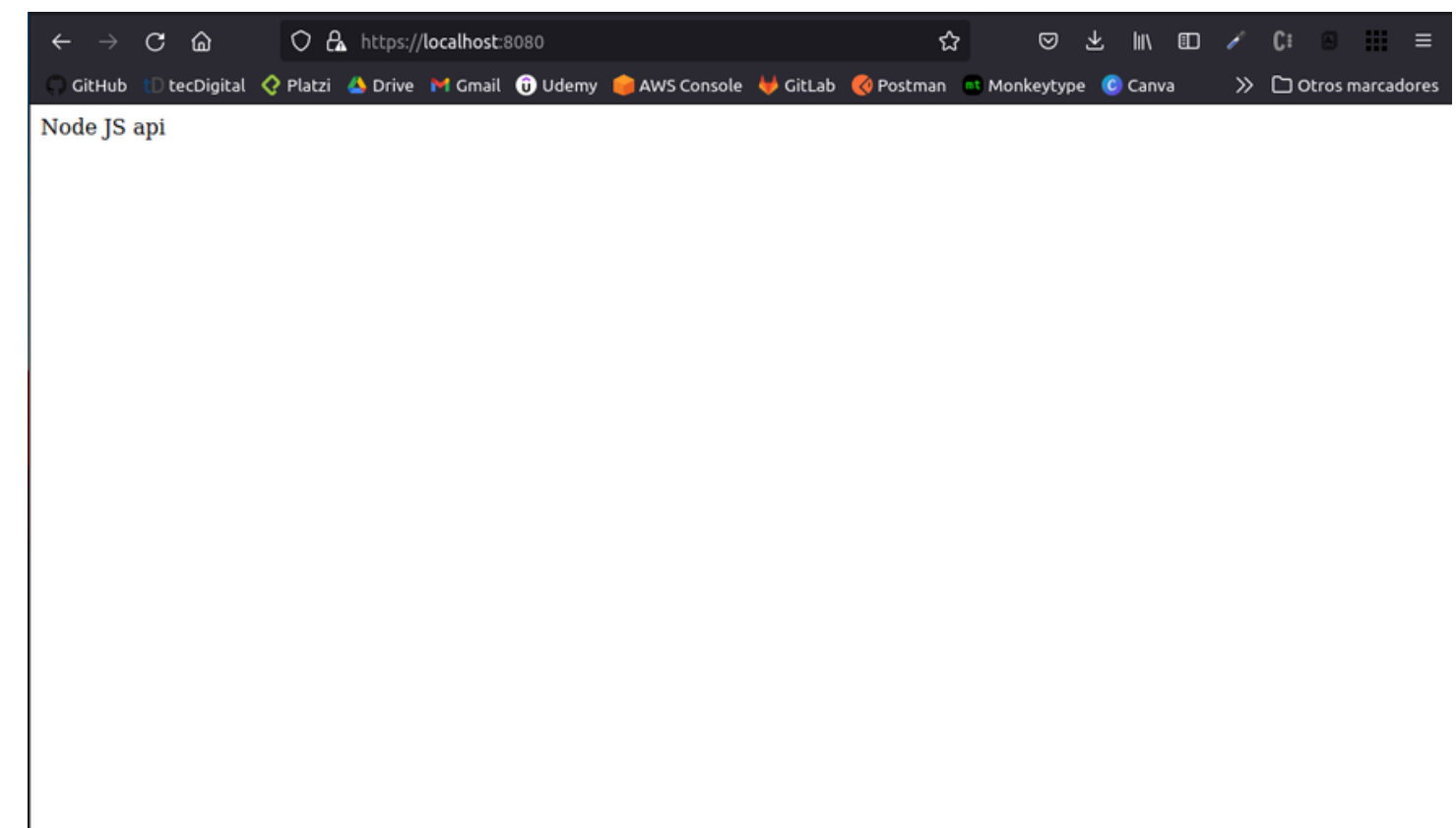
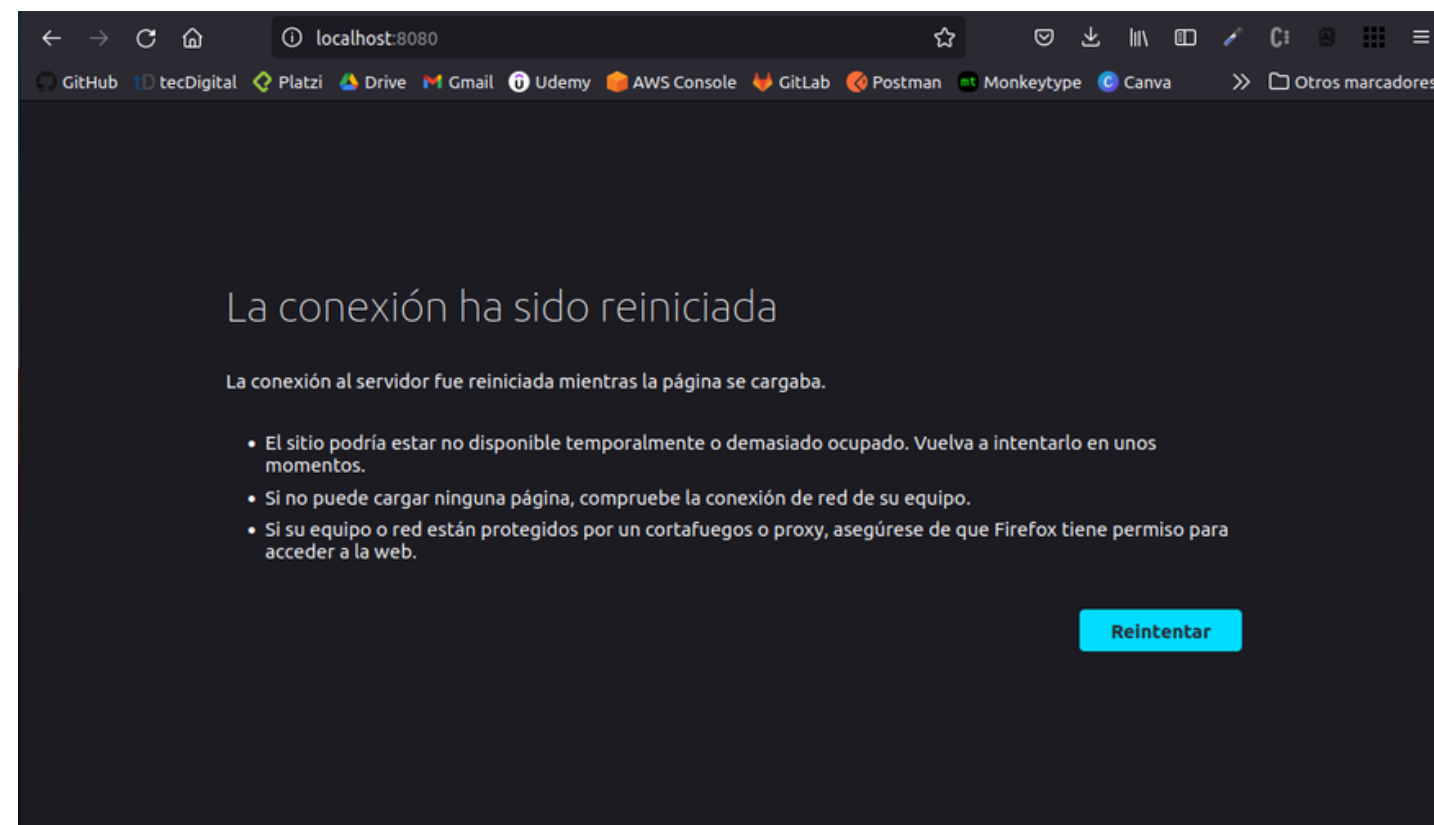
SE generaron las llaves y el certificado Self-Signed utilizando **openssl** y el modulo https de node js.

config

Luego de permitir que el navegador permita el certificado de nuestro API:

  <https://localhost:8080>

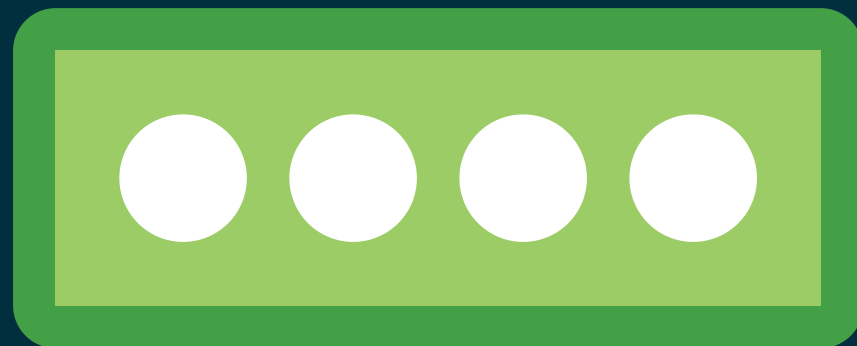
Realizamos una prueba para comprobar que ahora estamos con https. Al llamarlo con http la conexión falla. Cuando se llama con https nos conectamos y el navegador la acepta pero indica que es una conexión vulnerable ya que no tenemos el certificado instalado.



REST API Buenas Prácticas

Implementamos buenas prácticas en nuestro API:

- Paginación
- Filtros



Paginación

Se implementó la paginación en los métodos:

GET /api/spaces

GET /api/reservations

El API recibe mediante Query dos parámetros: offset y limit

Request

GET https://localhost:8080/api/spaces?offset=1&limit=2

Response

```
[
  {
    "id": 1,
    "state": "in-use",
    "detail": "handicapped-parking",
    "licensePlate": "8496321",
    "checkIn": "14:57",
    "reserved": true
  },
  {
    "id": 2,
    "state": "in-use",
    "detail": "indoor-parking",
    "licensePlate": "2696329",
    "checkIn": "14:57",
    "reserved": true
  },
  {
    "id": 3,
    "state": "in-use",
    "detail": "indoor-parking",
    "licensePlate": "8652855",
    "checkIn": "14:57",
    "reserved": true
  }
]
```

Empieza en el 1(offset) y da hasta 2(limit) respuestas adicionales.



REST API Buenas Prácticas

Implementamos buenas prácticas en nuestro API:

- Paginación
- Filtros



Filtros

Se implementó el filtro en los métodos:

GET /api/spaces

GET /api/reservations

El API recibe mediante Query el parámetro: filter

Request

GET

https://localhost:8080/api/spaces?filter=id,licensePlate,state

Response

```
[
  {
    "id": 1,
    "licensePlate": "8496321",
    "state": "in-use"
  },
  {
    "id": 2,
    "licensePlate": "2696329",
    "state": "in-use"
  },
  {
    "id": 3,
    "licensePlate": "8652855",
    "state": "in-use"
  },
]
```

Se solicitó que la respuesta incluyera únicamente:

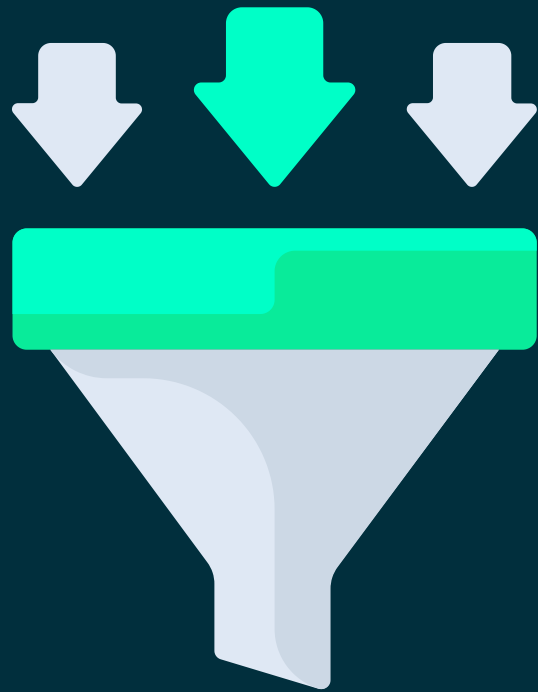
- id
- licensePlate
- state



REST API Buenas Prácticas

Implementamos buenas prácticas en nuestro API:

- Paginación
- Filtros



Filtros

Se implementó el filtro en los métodos:

GET /api/spaces

GET /api/reservations

El filtro recibe las propiedades que se desean ver en la respuesta.

Request

`https://localhost:8080/api/reservations?filter=checkIn,licensePlate,detail`

Response

```
[
  {
    "id": 1,
    "licensePlate": "8496321",
    "state": "in-use"
  },
  {
    "id": 2,
    "licensePlate": "2696329",
    "state": "in-use"
  },
]
```

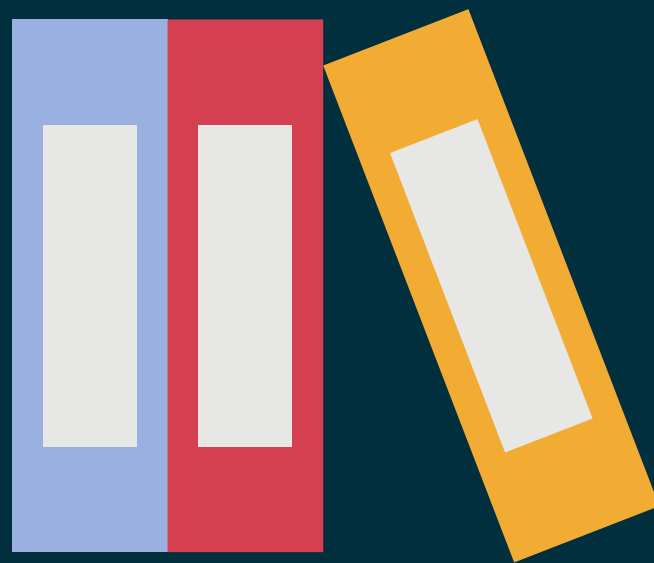
Se solicitó que filtrara los la respuesta incluyera únicamente:

- detail
- licensePlate
- checkIn



Docs

La documentación es un aspecto **demasiado importante**, es el medio por el que comunicamos a los desarrolladores cómo utilizar nuestro Rest API.



La documentación del API de Parking TEC se realizó utilizando la herramienta:



Swagger



Repositorio del proyecto

https://github.com/gaburolo/Laboratorio_REACT

Cliente React



Taller React

Availables

ID	STATE	DETAIL	RESERVED
4	free	indoor-parking	false
5	free	La cali-parking	false
6	free	La nave-parking	false

Reserved

ID	STATE	DETAIL	LICENSE PLATE	CHECK IN TIME	RESERVED
1	in-use	handicapped-parking	8496321	14:57	true
2	in-use	indoor-parking	2696329	14:57	true
3	in-use	indoor-parking	8652855	14:57	true

Add New Spaces

Description

Add new Space

Reserve Space

License Plate

Reserve Space

CE 5508

Parking-TEC

INVESTIGACION #1

Presentado por:
Esteban Alvarado
Martín Calderón
Olman Castro