# arcade

## Manual técnico

Primer semestre - mayo 2021

## INTRODUCCION

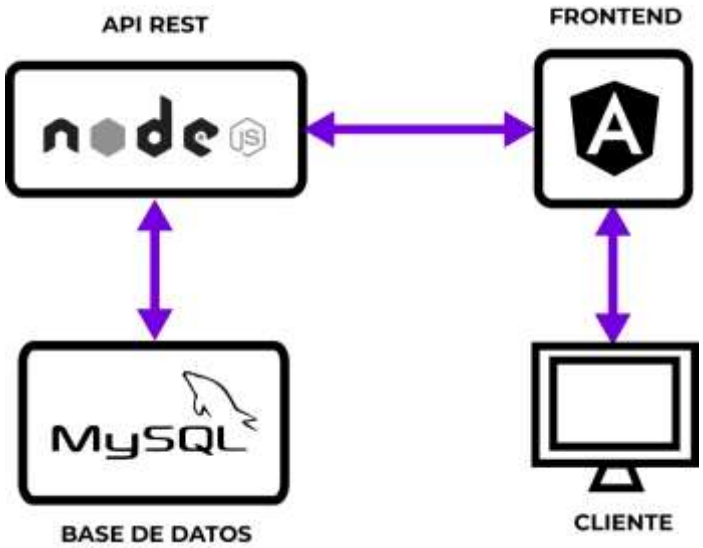
Se desarrolló una aplicación de tipo cliente – servidor cuyo objetivo es proporcionar un espacio para los amantes de los juegos retro, proporcionandoles esta red social con la que al registrarse pueden acceder a información sobre estos juegos, interactuar con otros usuarios a travéz de comentarios, además de cpntar con usuarios de tipo administrador que pueden modificar y eliminar los usuarios registrados además de agregar, modificar y eliminar juegos a la librería. Además de acceder a los reportes de los datos gestionados por la aplicación.

La capa del cliente de la aplicación fue desarrollada en Angular para darle una vista al cliente de lo que es nuestra aplicación, la capa del servidor fue desarrollada en NodeJS como un servicio de REST API con lo que se crearon los endpoints que serán consumidos en la capa del cliente. Toda la información de almacena en una base de datos desarrollada utilizando MySQL.

## DATOS TECNICOS

- Framework utilizado para la API REST: NodeJS
- Framework utilizado para el Frontend: Angular
- Software utilizado la base de datos: MySQL
- IDE utilizado: Visual Studio Code

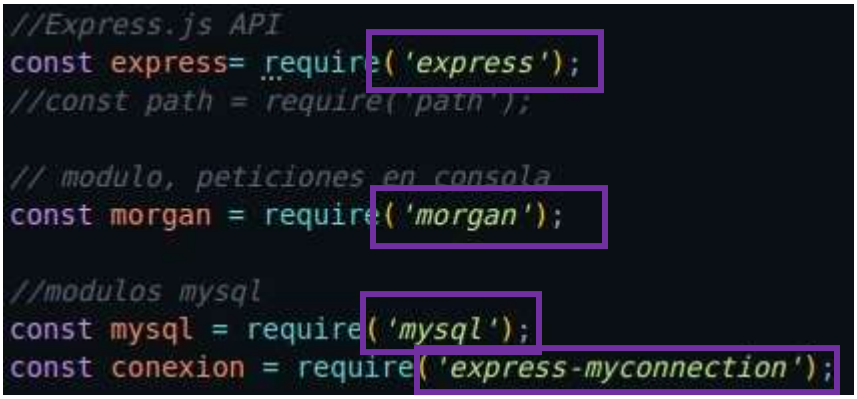
## ARQUITECTURA



**FLUJO DE TRABAJO**

1. El usuario accede al frontend de la aplicación a través de un navegador
2. El frontend al  detectar que el usuario ejecuta alguna de las acciones que este ofrece envía peticiones a la API REST.
3. La API REST al recibir una petición proveniente del frontend  que necesita obtener  datos de memoria o guardar datos en esta se comunica con la base de datos a través de el modulo de node.js llamado express, de esta forma se hace el intercambio de datos.
4. Con la respuesta de la base de datos la API REST puede responderle al frontend, y este a su vez le muestra respuesta al usuario.

**Módulos y conexión de base de datos con el servidor.**




Express: el framework express de Node.js para facilitar la lectura de los endpoints, y el uso de middlewares.

Morgan: nos permite visualizar las peticiones que realiza el usuario en consola indicando el tipo de solicitud (get, post, etc), el tiempo que se demora y la memoria que ha consumido la solicitud al servidor.

Mysql: fue utilizado como sistema para gestionar la base de datos.

Express-myconnection: nos permite enlazar la base de datos con el servidor.

**Ejemplo de la implementación de express-myconnection y morgan**

En el ejemplo anterior en el campo use: *'usuario_database'* se debe remplazar con el usuario de la base de datos que se quiera utilizar y en el campo password la contraseña del usuario para acceder a la base de datos.

**Controladores:** los controladores nos permiten estructurar de forma independiente las funciones que implementan los endpoints para poder dar una respuesta a las solicitadas por el usuario, permitiendo hacer referencia a ellos en las peticiones correspondientes a cada ruta.

Esquema de los controladores:

```
const controlador = {};

controlador.crear = (req, res) => {
    res.send('agregar juego')
}

controlador.eliminar = (req, res) =>{
    res.send('eliminar juego')
}

controlador.actualizar = (req, res) =>{
    res.send('actualizar juego')
}

controlador.vista = (req, res) =>{
    res.send('ver juegos')
}

module.exports = controlador;
```

Ejemplo de un controlador en este ejemplo se muestra la estructura de la función que se implementa cuando el usuario inicia sesión en la aplicación.

```
controller.entrar = (req , res) =>{
  const data = req.body;

  req.getConnection((err,conn)=>{
     conn.query('SELECT * FROM usuarios',(err,usuarios)=>{
         if (err){
             res.json(err);
         }
         var us =""
         for(var i = 0 ; i < usuarios.length;i++){
          if (usuarios[i].NoUsuario == data.NoUsuario){
            if (usuarios[i].Contra == data.Contra){
              res.redirect('http://localhost:4200/perfil');
            }
            else{
              res.redirect('http://localhost:4200/login');
            }
          }
         }
         res.redirect('http://localhost:4200/login');
     });

  });

};
```

Rutas: son las que implementan la función contenida en los controladores, por lo cual son las que se ejecutan al momento de que el usuario realiza una solicitud.

Ejemplo de la estructura de las rutas.

```
const express = require('express');
const router = express.Router();

const juegoControlador = require('../controladores/juegoControlador');

// --------------Endpoints JUEGOS-----------------
router.get('/create', juegoControlador.crear);
router.get('/delete', juegoControlador.eliminar);
router.get('/update', juegoControlador.actualizar);
router.get('/gamesList', juegoControlador.vista);

module.exports = router;
```

Para continuar con el ejemplo propuesto en los controladores se muestran las rutas que implementa el usuario.

```
const express = require('express');
const router = express.Router();

const customerController = require("../contollers/customerControlers")

router.get('/', customerController.list);
router.post('/Crear',customerController.guardar)
router.post('/entrar',customerController.entrar)
```