|  |  |
| --- | --- |
| **Nombre:** | Luis de la Garza González |
| **Matrícula:** | al03101869 |
| **Nombre del curso:** | Backend con Node.js |
| **Trabajo:** | Proyecto Final |
| **Nombre del profesor:** | Julio César Cruz Cruz |
| **Fecha:** | 16.12.23 |

**Contenido**

[**Objetivo** 3](#_Toc153594851)

[**Descripción** 3](#_Toc153594852)

[**Avance 1** 3](#_Toc153594853)

[**Avance 2** 4](#_Toc153594854)

[**Avance 3** 13](#_Toc153594855)

[**Avance 4** 17](#_Toc153594856)

[**Avance 5** 18](#_Toc153594857)

[**Conclusiones** 29](#_Toc153594858)

[Checklist del proyecto final 30](#_Toc153594859)

**Ilustraciones:**

[Ilustración 1 - Base de datos libros y sus tablas autores y libros. 4](https://d.docs.live.net/5365f7fcd4da17b0/Tec%20Milenio/Back%20end%20nivel%20intermedio/Proyecto%20final%20-%20Back%20end%20nivel%20intermedio%20v%20231109%2009.56%20.docx#_Toc150430518)

[Ilustración 2 - Creación de la base de datos “Libros” 6](https://d.docs.live.net/5365f7fcd4da17b0/Tec%20Milenio/Back%20end%20nivel%20intermedio/Proyecto%20final%20-%20Back%20end%20nivel%20intermedio%20v%20231109%2009.56%20.docx#_Toc150430519)

[Ilustración 3 - Creación de la tabla “libros”. 6](https://d.docs.live.net/5365f7fcd4da17b0/Tec%20Milenio/Back%20end%20nivel%20intermedio/Proyecto%20final%20-%20Back%20end%20nivel%20intermedio%20v%20231109%2009.56%20.docx#_Toc150430520)

[Ilustración 4 - Creación de la tabla "autores" 7](https://d.docs.live.net/5365f7fcd4da17b0/Tec%20Milenio/Back%20end%20nivel%20intermedio/Proyecto%20final%20-%20Back%20end%20nivel%20intermedio%20v%20231109%2009.56%20.docx#_Toc150430521)

[Ilustración 5 - Tabla libros con 40 registros 8](https://d.docs.live.net/5365f7fcd4da17b0/Tec%20Milenio/Back%20end%20nivel%20intermedio/Proyecto%20final%20-%20Back%20end%20nivel%20intermedio%20v%20231109%2009.56%20.docx#_Toc150430522)

[Ilustración 6 - Tabla autores con registros 10](https://d.docs.live.net/5365f7fcd4da17b0/Tec%20Milenio/Back%20end%20nivel%20intermedio/Proyecto%20final%20-%20Back%20end%20nivel%20intermedio%20v%20231109%2009.56%20.docx#_Toc150430523)

[Ilustración 7 - Resultado de la conexión y consulta a la base de datos. 15](https://d.docs.live.net/5365f7fcd4da17b0/Tec%20Milenio/Back%20end%20nivel%20intermedio/Proyecto%20final%20-%20Back%20end%20nivel%20intermedio%20v%20231109%2009.56%20.docx#_Toc150430524)

[Ilustración 8 - Selección de un registro específico. 16](https://d.docs.live.net/5365f7fcd4da17b0/Tec%20Milenio/Back%20end%20nivel%20intermedio/Proyecto%20final%20-%20Back%20end%20nivel%20intermedio%20v%20231109%2009.56%20.docx#_Toc150430525)

[Ilustración 9 - Prueba de la API de Abstract 17](https://d.docs.live.net/5365f7fcd4da17b0/Tec%20Milenio/Back%20end%20nivel%20intermedio/Proyecto%20final%20-%20Back%20end%20nivel%20intermedio%20v%20231109%2009.56%20.docx#_Toc150430526)

[Ilustración 10 - Base de datos "registro" 19](https://d.docs.live.net/5365f7fcd4da17b0/Tec%20Milenio/Back%20end%20nivel%20intermedio/Proyecto%20final%20-%20Back%20end%20nivel%20intermedio%20v%20231109%2009.56%20.docx#_Toc150430527)

[Ilustración 11 - Registros insertados incluyendo la hora. 22](https://d.docs.live.net/5365f7fcd4da17b0/Tec%20Milenio/Back%20end%20nivel%20intermedio/Proyecto%20final%20-%20Back%20end%20nivel%20intermedio%20v%20231109%2009.56%20.docx#_Toc150430528)

# **Objetivo**

Generar un portafolio de evidencias que integre el código, vídeos y capturas de pantalla, las cuales ilustran cada paso y ejecución que fue realizado para la entrega del avance, esto en conjunto con sus respectivas reflexiones.

# **Descripción**

Comprender la estructura y funcionamiento de un back end en desarrollo web, utilizando el lenguaje de programación adecuado, así como las bases de datos necesarias para su implementación de forma local.

# **Avance 1**

|  |  |
| --- | --- |
| **Entregables** | **Realizado** |
| **Fase 1: Creación de un servidor básico** |  |
| Entorno de Node.js instalado y configurado | ✔ |
| Servidor básico creado y funcionando | ✔ |
| Implementación y demostración de Event Loop y asincronía | ✔ |
| Verificación de la conexión del servidor | ✔ |

**Entorno de Node.js instalado y configurado:**

Podemos ver en la siguiente imagen que Node.js se encuentra instalado en Visual Studio Code:

Ilustración 1 - Entorno de Node.js instalado y configurado.

**Texto

Descripción generada automáticamente**

**Servidor básico creado y funcionando**

|  |
| --- |
| var http = require('http');  //create a server object:  http.createServer(function (req, res) {  res.writeHead(200, { 'Content-Type': 'text/html' });  res.write('Hello World!'); //write a response to the client  res.end(); //end the response  }).listen(8080); //the server object listens on port 8080 |

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente**Verificación de la conexión del servidor**

Ilustración 2 – Verificación de la conexión del servidor.

Nota. - la creación de las tablas se muestra en el siguiente capítulo.

**Implementación y demostración de Event Loop y asincronía**

El siguiente código implementa y demuestra es funcionamiento del Event Loop mediante una función asíncrona la cual toma en ejecutarse ts segundos, pero sin bloquear la ejecución del código restante.

|  |
| --- |
| // Ejemplo de una función asíncrona  const ts = 3; // tiempo en segundos que va a ocupar el setTimeout()  function makePromise(x) {  return new Promise(resolve => {  setTimeout(() => { resolve(x)}, x \* 1000);  });  }    async function asyncFunc() {  var x = makePromise(ts);  console.log('inicio');  return x;  }    const returnedProm = asyncFunc(); // la función asíncrona retorna una promesa.  returnedProm.then((x) => console.log(`Fin en ${x\*1000} ms`));  // La promesa es cumplida con el valor retornado desde la función asíncrona,  // entonces esta línea despliega los mili segundos que la función tardó en  // cumplir su promesa. |

El resultado de la ejecución del código es el siguiente.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Podemos ver que, aunque la llamada a la función makePromise(x) se encuentra en el código antes que el console.log('inicio'), este console.log se despliega antes que el console.log(`Fin en ${x\*1000} ms`) el cual es ejecutado al cumplirse la promesa. De esta manera la función asíncrona no bloquea la ejecución del programa.

El evento loop es el encargado de coordinar la ejecución de los eventos, incluyendo el renderizado de la página Web y las colas de tareas de los eventos.

# **Avance 2**

|  |  |
| --- | --- |
| **Fase 2: Desarrollo de una aplicación web segura** |  |
| Código organizado en módulos | ✔ |
| Manejo de rutas y peticiones HTTP con Express | ✔ |
| Implementación de autenticación y autorización | ✔ |
| Prevención de ataques comunes (e.g. Cross-Site Scripting, Inyección SQL) |  |

**Código organizado en módulos**

**Manejo de rutas y peticiones HTTP con Express**

**Implementación de autenticación y autorización**

Para manejar sesiones en Node.js con Express, se puede utilizar el paquete express-session. Este paquete permite almacenar datos de sesión en el servidor y acceder a ellos en cada solicitud.

Instalemos el paquete express-session usando el siguiente comando en la terminal:

|  |
| --- |
| npm install express-session |

**Prevención de ataques comunes (e.g. Cross-Site Scripting, Inyección SQL)**

Captura de pantalla de computadora

Descripción generada automáticamenteLa siguiente imagen muestra la utilización de los comandos npm ls y npm Audit.

Npm ls lista todos los paquetes desde el directorio padre de manera que podamos verificar las dependencias, si es necesaria la actualización de algún paquete, así como las vulnerabilidades conocidas de los mismos.

npm audit es un comando que analiza el proyecto en busca de vulnerabilidades y proporciona un informe de vulnerabilidades conocidas. El informe devuelto incluye instrucciones sobre cómo actuar sobre esta información. El comando se cerrará con un código de salida 0 si no se encontraron vulnerabilidades.

**Inyección SQL (SQL injection).**

La prevención de la inyección SQL es un aspecto crucial de la seguridad de las aplicaciones web. En Node.js, puede evitar la inyección de código SQL mediante instrucciones preparadas (prepared statements) y consultas con parámetros (parameterized queries) Las instrucciones preparadas y las consultas parametrizadas separan el código SQL de la entrada del usuario, lo que dificulta que los atacantes inyecten código SQL malicioso en la aplicación.

La biblioteca node-mysql, que es un conector MySQL popular para Node.js, realiza automáticamente el escape cuando se usa como ya lo está haciendo. También se puede usar el método mysql.escape() para escapar de los valores de consulta cuando los valores de consulta son variables proporcionadas por el usuario.

Aquí hay algunos consejos adicionales para evitar la inyección SQL en Node.js:

* No permitir varias declaraciones.
* Utilizar marcadores de posición en lugar de interpolación de variables.
* Validar la entrada del usuario.
* Entrada del usuario en la lista de permitidos.

1. **Crea una base de datos en MySQL que se llame: Libros.**
2. **Crea una tabla que se llame: Libros y que contenga los siguientes campos:** 
   1. **Título.**
   2. **Páginas.**
   3. **Fecha de publicación.**
   4. **Editorial.**
   5. **Primary key.**
3. **Crea una tabla que se llame: Autores y que contenga los siguientes campos:** 
   1. **Nombre.**
   2. **Apellidos.**
   3. **Primary Key.**
4. **Llena las tablas con 40 registros.**
5. **Deberás realizar capturas de pantalla de cada uno de los pasos y guardarlos en tu documento que empezaste a organizar, recuerda que las capturas de pantalla deben de estar organizadas con número y título.**

**Consideraciones generales**

**Para el aprendedor:**

* **La Primary Key es crucial para crear la relación 1 a muchos.**
* **Ambas tablas deben contar con al menos 40 registros.**

**Para el instructor:**

* **Ambas tablas deben contar con un Primary Key.**
* **Revisar que la relación de ambas tablas sea 1 a muchos.**
* **Ambas tablas deben tener al menos 40 registros.**

1.- Creación de la base de datos “Libros”:

Texto

Descripción generada automáticamente

Creación de la tabla **libros**:

Texto

Descripción generada automáticamente

Ilustración 3 Creación de la tabla “libros”.

Creación de la tabla autores:

Texto

Descripción generada automáticamente

Ilustración 4 - Creación de la tabla "autores"

Se insertan 40 registros en cada tabla (**libros** y **autores**)

|  |
| --- |
| INSERT INTO Libros (titulo, paginas, f\_publicacion, editorial) VALUES  (' Cien años de soledad',238, '2023/10/26','Acantilado'),  (' El señor de los anillos (Trilogía)',240, '2023/04/29','Aguilar'),  (' 1984',201, '2022/10/31','Akal'),  (' Un mundo feliz',364, '2022/05/04','Alba'),  (' Orgullo y prejuicio',101, '2021/11/05','Alfaguara'),  (' Crimen y castigo',500, '2021/05/09','Alianza'),  (' Lolita',373, '2020/11/10','Almadía'),  (' Ulises',149, '2020/05/14','Anagrama'),  (' Madame Bovary',339, '2019/11/16','Acantilado'),  (' En busca del tiempo perdido',148, '2019/05/20','Aguilar'),  (' Don Quijote de la Mancha',364, '2018/11/21','Akal'),  (' El retrato de Dorian Gray',204, '2018/05/25','Alba'),  (' Ana Karenina',153, '2017/11/26','Alfaguara'),  (' El Principito',134, '2017/05/30','Alianza'),  (' El proceso',280, '2016/12/01','Almadía'),  (' El ruido y la furia',155, '2016/06/04','Anagrama'),  (' Hamlet',197, '2015/12/07','Acantilado'),  (' Lo que el viento se llevó',450, '2015/06/10','Aguilar'),  (' La Odisea',359, '2014/12/12','Akal'),  (' Las uvas de la ira',176, '2014/06/15','Alba'),  (' El guardián entre el centeno',172, '2013/12/17','Alfaguara'),  (' Cumbres borrascosas',105, '2013/06/20','Alianza'),  (' El gran Gatsby',130, '2012/12/22','Almadía'),  (' Mil soles espléndidos',273, '2012/06/25','Anagrama'),  (' Alicia en el país de las maravillas',272, '2011/12/28','Acantilado'),  (' Rebelión en la granja',242, '2011/07/01','Aguilar'),  (' Los pilares de la tierra',382, '2011/01/02','Akal'),  (' Guerra y paz',455, '2010/07/06','Alba'),  (' Memorias de una geisha',496, '2010/01/07','Alfaguara'),  (' Frankenstein',232, '2009/07/11','Alianza'),  (' Los viajes de Gulliver',180, '2009/01/12','Almadía'),  (' La ladrona de libros',322, '2008/07/16','Anagrama'),  (' Matar a un ruiseñor',430, '2008/01/18','Acantilado'),  (' El conde de Montecristo',194, '2007/07/22','Aguilar'),  (' Los juegos del hambre',391, '2007/01/23','Akal'),  (' Harry Potter y la piedra filosofal',190, '2006/07/27','Alba'),  (' El señor de las moscas',408, '2006/01/28','Alfaguara'),  (' Moby Dick',266, '2005/08/01','Alianza'),  (' Los miserables',407, '2005/02/02','Almadía'),  (' Las aventuras de Huckleberry Finn',197, '2004/08/06','Anagrama'); |

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 5 - tabla libros con 40 registros

|  |
| --- |
| INSERT INTO Autores (nombre, apellidos) VALUES  ('Gabriel','García Márquez'),  ('J.R.R.','Tolkien'),  ('George','Orwell'),  ('Aldous','Huxley'),  ('Jane','Austen'),  ('Fiódor','Dostoyevski'),  ('Vladimir','Nabokov'),  ('James','Joyce'),  ('Gustave','Flaubert'),  ('Marcel','Proust'),  ('Miguel','de Cervantes'),  ('Oscar','Wilde'),  ('León','Tolstói'),  ('Antoine','de Saint-Exupéry'),  ('Franz','Kafka'),  ('William','Faulkner'),  ('William','Shakespeare'),  ('Margaret','Mitchell'),  ('Homero','S/A'),  ('John','Steinbeck'),  ('J.','D. Salinger'),  ('Emily','Brontë'),  ('F.','Scott Fitzgerald'),  ('Khaled','Hosseini'),  ('Lewis','Carroll'),  ('George','Orwell'),  ('Ken','Follett'),  ('León','Tolstói'),  ('Arthur','Golden'),  ('Mary','W'),  ('Jonathan','Swift'),  ('Markus','Zusak'),  ('Harper','Lee'),  ('Alejandro','Dumas'),  ('Suzanne','Collins'),  ('J.K.','Rowling'),  ('William','Golding'),  ('Herman','Melville'),  ('Victor','Hugo'),  ('Mark','Twain'); |

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 6 - tabla autores con registros

|  |
| --- |
|  |
|  |

|  |
| --- |
| **Relacionando la tabla libros con la tabla autores:**  Nos aseguramos de que el campo **autor** esté validado con la tabla de **autores**: |
| ALTER TABLE libros ADD autor int;  ALTER TABLE libros ADD CONSTRAINT autor FOREIGN KEY (autor) REFERENCES autores(id); |
|  |
|  |
| De esta manera un autor puede tener varios libros. |
| Para poder tener una relación uno a muchos en ambos sentidos, debemos utilizar una tabla intermedia que llamaremos relación y que contendrá los pares libro – autor, de esa manera un libro puede tener varios autores y un autor puede tener varios libros: |
|  |

**Lenguajes de programación Back End:**

**Python**

Desde que Python fue creado en 1991 ha crecido hasta convertirse en uno de los lenguajes de programación multipropósito líderes y es el lenguaje de programación de backend que los desarrolladores esperan aprender.

Características

* Relativamente fácil de aprender: uno de los beneficios de Python es el estilo de programación similar al inglés que lo hace altamente legible. Por lo tanto, programar y leer códigos Python es relativamente fácil.
* Python disfruta del soporte de bibliotecas muy amplias que reducen la necesidad de escribir códigos manualmente. Algunas bibliotecas contienen códigos, entre otros, para correo electrónico, navegación, asignaciones de bases de datos, pruebas unitarias y mucho más.
* Python es una plataforma de código abierto de descarga gratuita.
* Códigos integrables: con la regla “Escribe una vez, ejecútalo en cualquier lugar” (WORA, del inglés “Write Once Run Anywhere”), el código Python se puede incrustar en el código fuente de otros lenguajes como C++.

Limitaciones

* La ejecución del código Python se vuelve lenta cuando se interrumpe. Este déficit perjudica a todo el proyecto de desarrollo de aplicaciones.
* La capa de acceso a la base de datos de Python está menos desarrollada en comparación con otros lenguajes de programación backend.
* Se requieren pruebas exhaustivas para detectar errores dentro de los códigos de Python.
* Depende en gran medida de bibliotecas y Frame Works de terceros.

|  |
| --- |
|  |

**Java**

Java es uno de los lenguajes de programación más influyentes en la actualidad, el índice TIOBE de 2021 clasifica a Java en segundo lugar entre las tecnologías de desarrollo de backend.

A lo largo de los años, Java se ha convertido en la plataforma elegida por los desarrolladores que prefieren crear aplicaciones web con funciones personalizadas. Java también es útil para el desarrollo móvil y las aplicaciones en dispositivos integrados.

Características

* Escalable y sencillo: es útil para crear aplicaciones escalables al permitir que el servidor ejecute varias instancias. Los componentes de Java están fácilmente disponibles y la sintaxis es fácil de entender. Todas estas características hacen que sea sobresaliente como tecnología de desarrollo backend.
* Multihilo: Java es capaz de manejar solicitudes en hilos independientes, por eso que Java funciona muy bien con aplicaciones que requieren gran potencia de procesamiento.
* Bibliotecas de código abierto masivas.
* Seguridad. Java es conocido por su estricta seguridad; proporciona varias características que mitigan los riesgos. Por ejemplo, la máquina virtual de Java comprueba los códigos de bytes de Java para mantener a raya a los virus. Otras características que mejoran la seguridad son el modelo de seguridad de Java y la prueba de código reutilizable.

Limitaciones

* La programación en Java requiere mucho tiempo.
* La programación de bajo nivel y el comando para la recolección de basura están notoriamente ausentes.
* Los grandes requisitos de hardware hacen que la implementación de backend con tecnología Java sea costosa.

**Conclusión:**

Para el presente trabajo se selecciona **Java** como el lenguaje de programación para el back end ya que es muy conocido y cuenta con un amplio soporte, aunque en un futuro cercano es muy importante el aprender a programar en **Python** ya que también permite desarrollar aplicaciones de Big Data y de inteligencia artificial.

# **Avance 3**

|  |  |
| --- | --- |
| **Fase 3: Integración de bases de datos y despliegue** |  |
| Integración exitosa con la base de datos seleccionada | ✔ |
| Consultas a bases de datos y manejo de datos | ✔ |
| Configuración de entornos (producción, desarrollo y pruebas) | ✔ |
| Pruebas unitarias y de integración ejecutadas y pasadas |  |

Se utiliza la base de datos MySQL llamada “libros”.

Crea una tabla que se llame: Libros y que contenga los siguientes campos:

* id
* Título.
* Páginas.
* Fecha de publicación.
* Editorial.
* Primary key.

Crea una tabla que se llame: Autores y que contenga los siguientes campos:

* id
* Nombre.
* Apellidos.
* Primary Key.

|  |
| --- |
| const mysql = require('mysql');  const connection = mysql.createConnection({  localhost: 'localhost',  user: 'root',  password: 'Luis0908',  database: 'libros',  port: 3306  })  const autor = {  nombre: 'luis',  apellidos: 'de la Garza González'  }  const query = connection.query('insert into autores set ?', autor, (err, result) => {  if (err) throw err;  console.log(result);  });  connection.query('select \* from autores', (err, rows) => {  if (err) throw err;  console.log('Datos recibidos de la base de datos: ');  console.log(rows);  });  // connection.query('update users set email = ? where id = ?', ['luisgg121@gmail.com', 2], (err, rows) => {  // if (err) throw err;  // console.log(`${rows.changedRows} Registros cambiados.`);  // });  // connection.query('delete from autores where id = ?', [1], (err, rows) => {  // if (err) throw err;  // console.log(`Los registro se eliminaron correctamente.`);  // }); connection.close();  } catch (Exception exception) {  System.out.println(exception);  }  } // function ends  } // class ends |
|  |
| Se ejecuta el programa en Visual Studio Code  $ node server.js  OkPacket {  fieldCount: 0,  affectedRows: 1,  insertId: 41,  serverStatus: 2,  warningCount: 0,  message: '',  protocol41: true,  changedRows: 0  }  Datos recibidos de la base de datos:  [  RowDataPacket {  id: 1,  nombre: 'Gabriel',  apellidos: 'García Márquez'  },  RowDataPacket { id: 2, nombre: 'J.R.R.', apellidos: 'Tolkien' },  RowDataPacket { id: 3, nombre: 'George', apellidos: 'Orwell' },  RowDataPacket { id: 4, nombre: 'Aldous', apellidos: 'Huxley' },  RowDataPacket { id: 5, nombre: 'Jane', apellidos: 'Austen' },  RowDataPacket { id: 6, nombre: 'Fiódor', apellidos: 'Dostoyevski' },  RowDataPacket { id: 7, nombre: 'Vladimir', apellidos: 'Nabokov' },  RowDataPacket { id: 8, nombre: 'James', apellidos: 'Joyce' },  RowDataPacket { id: 9, nombre: 'Gustave', apellidos: 'Flaubert' },  RowDataPacket { id: 10, nombre: 'Marcel', apellidos: 'Proust' },  RowDataPacket { id: 11, nombre: 'Miguel', apellidos: 'de Cervantes' },  RowDataPacket { id: 12, nombre: 'Oscar', apellidos: 'Wilde' },  RowDataPacket { id: 13, nombre: 'León', apellidos: 'Tolstói' },  RowDataPacket {  id: 14,  nombre: 'Antoine',  apellidos: 'de Saint-Exupéry'  },  RowDataPacket { id: 15, nombre: 'Franz', apellidos: 'Kafka' },  RowDataPacket { id: 16, nombre: 'William', apellidos: 'Faulkner' },  RowDataPacket { id: 17, nombre: 'William', apellidos: 'Shakespeare' },  RowDataPacket { id: 18, nombre: 'Margaret', apellidos: 'Mitchell' },  RowDataPacket { id: 19, nombre: 'Homero', apellidos: 'S/A' },  RowDataPacket { id: 20, nombre: 'John', apellidos: 'Steinbeck' },  RowDataPacket { id: 21, nombre: 'J.', apellidos: 'D. Salinger' },  RowDataPacket { id: 22, nombre: 'Emily', apellidos: 'Brontë' },  RowDataPacket { id: 23, nombre: 'F.', apellidos: 'Scott Fitzgerald' },  RowDataPacket { id: 24, nombre: 'Khaled', apellidos: 'Hosseini' },  RowDataPacket { id: 25, nombre: 'Lewis', apellidos: 'Carroll' },  RowDataPacket { id: 26, nombre: 'George', apellidos: 'Orwell' },  RowDataPacket { id: 27, nombre: 'Ken', apellidos: 'Follett' },  RowDataPacket { id: 28, nombre: 'León', apellidos: 'Tolstói' },  RowDataPacket { id: 29, nombre: 'Arthur', apellidos: 'Golden' },  RowDataPacket { id: 30, nombre: 'Mary', apellidos: 'W' },  RowDataPacket { id: 31, nombre: 'Jonathan', apellidos: 'Swift' },  RowDataPacket { id: 32, nombre: 'Markus', apellidos: 'Zusak' },  RowDataPacket { id: 33, nombre: 'Harper', apellidos: 'Lee' },  RowDataPacket { id: 34, nombre: 'Alejandro', apellidos: 'Dumas' },  RowDataPacket { id: 35, nombre: 'Suzanne', apellidos: 'Collins' },  RowDataPacket { id: 36, nombre: 'J.K.', apellidos: 'Rowling' },  RowDataPacket { id: 37, nombre: 'William', apellidos: 'Golding' },  RowDataPacket { id: 38, nombre: 'Herman', apellidos: 'Melville' },  RowDataPacket { id: 39, nombre: 'Victor', apellidos: 'Hugo' },  RowDataPacket { id: 40, nombre: 'Mark', apellidos: 'Twain' },  RowDataPacket {  id: 41,  nombre: 'luis',  apellidos: 'de la Garza González'  }  ] |
|  |
|  |

1. **Crea una base de datos en MySQL que se llame: Libros.**
2. **Crea una tabla que se llame: Libros y que contenga los siguientes campos:** 
   1. **Título.**
   2. **Páginas.**
   3. **Fecha de publicación.**
   4. **Editorial.**
   5. **Primary key.**
3. **Crea una tabla que se llame: Autores y que contenga los siguientes campos:** 
   1. **Nombre.**
   2. **Apellidos.**
   3. **Primary Key.**
4. **Llena las tablas con 40 registros.**
5. **Deberás realizar capturas de pantalla de cada uno de los pasos y guardarlos en tu documento que empezaste a organizar, recuerda que las capturas de pantalla deben de estar organizadas con número y título.**

**Consideraciones generales**

**Para el aprendedor:**

* **La Primary Key es crucial para crear la relación 1 a muchos.**
* **Ambas tablas deben contar con al menos 40 registros.**

**Para el instructor:**

* **Ambas tablas deben contar con un Primary Key.**
* **Revisar que la relación de ambas tablas sea 1 a muchos.**
* **Ambas tablas deben tener al menos 40 registros.**

# **Avance 4**

|  |  |
| --- | --- |
| **Fase 4: Implementación de microservicios y optimización** |  |
| Implementación de contenedores (Docker) y microservicios |  |
| Despliegue de microservicios en entornos de producción |  |
| Aplicación de estrategias de optimización en Node.js |  |
| Uso de herramientas de monitoreo, profiling y debugging |  |

# **Avance 5**

|  |  |
| --- | --- |
| **Fase Final: Integración y entrega** |  |
| Integración exitosa de todos los componentes del proyecto |  |
| Documentación completa, incluyendo comentarios en el código y guías de usuario |  |
| Presentación o demostración de la aplicación completa y escalable |  |
| Código fuente y otros archivos relevantes entregados en el formato requerido |  |

1.- Creación de la base de datos “Libros”:

|  |
| --- |
| 1. **De acuerdo con el lenguaje de programación elegido crea el código que te permita conectar la base de datos con el back end.**     1. **Conectarás la base de datos “Libros” usando el lenguaje que hayas elegido.** 2. **Crea el código que te permita obtener todos los registros de la base de datos.**     1. **Utiliza el Query SELECT después de establecer la conexión con la base de datos.** 3. **Crea el código que te permita obtener un registro en específico.**     1. **Utiliza el Query SELECT FROM después de establecer la conexión con la base de datos.** 4. **Deberás realizar capturas de pantalla de cada uno de los pasos y guardarlos en tu archivo Word.**   **Consideraciones generales**  **Para el aprendedor**   * **La conexión debe ser exitosa para que el Query funcione.** * **Debes establecer primero la conexión antes de consultar la información.**   **Para el instructor**   * **Las capturas de pantalla deben mostrar el código utilizado y el resultado de este al** * **momento de su ejecución.** * **El alumno debe establecer primero la conexión con la base de datos.** * **La conexión debe ser exitosa para que el alumno pueda consultar la información.** |

**Código en Java para conectar la base de datos con el back end.**

|  |
| --- |
| package src;  import java.sql.\*;  public class GFG {  public static void main(String arg[]) {  Connection connection = null;  try {  // las siguientes dos líneas son usadas para la conectividad.  Class.forName("com.mysql.cj.jdbc.Driver");  connection = DriverManager.getConnection(  "jdbc:mysql://localhost:3306/libros",  "root", "admin2023");  Statement statement;  statement = connection.createStatement();  ResultSet resultSet;  resultSet = statement.executeQuery(  "select \* from libros");  int id;  String titulo;  while (resultSet.next()) {  id = resultSet.getInt("id");  titulo = resultSet.getString("titulo").trim();  System.out.println("Id : " + id  + " Titulo : " + titulo);  }  resultSet.close();  statement.close();  connection.close();  } catch (Exception exception) {  System.out.println(exception);  }  } // function ends  } // class ends |
|  |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
| **Resultado** |
| Ilustración 7 - resultado de la conexión y consulta a la base de datos. |

Para obtener un registro en específico simplemente modificamos el SELECT para incluir la condicionante WHERE y especificamos campos y valores buscados, por ejemplo:

|  |
| --- |
| resultSet = statement.executeQuery(  "select \* from libros where titulo = 'Lolita'"); |

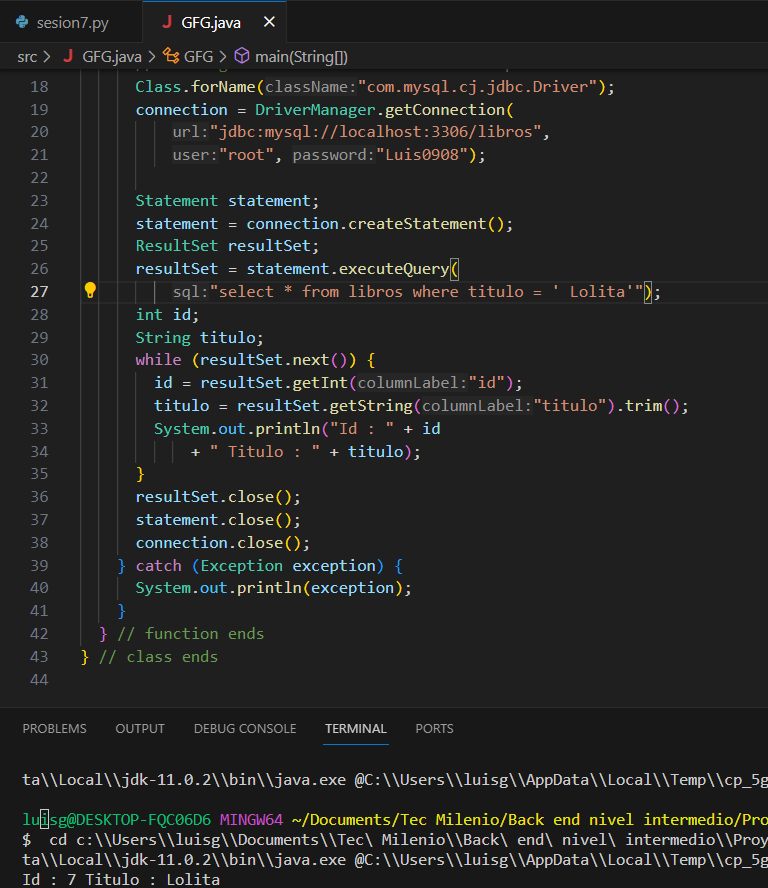


Ilustración 8 - selección de un registro específico.

|  |
| --- |
|  |

1. **Realiza en internet la búsqueda de una API (ubicación) y copia el código.**

**API de geolocalización de IP**

La API de geolocalización IP de **Abstract** es una API JSON rápida, ligera, moderna y RESTful, que se utiliza para determinar la ubicación y otros detalles de las direcciones IP en más de 190 países.

* Primary Key:675751b331544912a78c83426eef4445
* https://[ipgeolocation.abstractapi.com/v1/?api\_key=675751b331544912a78c83426eef4445](https://ipgeolocation.abstractapi.com/v1/?api_key=675751b331544912a78c83426eef4445)

Texto

Descripción generada automáticamentePrimero comprobamos que la API devuelva los resultados esperados:

Ilustración 9 - prueba de la API de Abstract

1. **Código en JavaScript para obtener la ubicación mediante una API.**

|  |
| --- |
| const options = { method: 'GET' };  fetch('https://ipgeolocation.abstractapi.com/v1/?api\_key=675751b331544912a78c83426eef4445', options)  .then(response => response.json())  .then(response => {  console.log(response);  const latitud = response.latitude, longitud = response.longitude;  const ip = response.ip\_address;  console.log("Tu IP es: ", ip);  console.log("Tus coordenadas son: ", latitud, longitud);  })  .catch(err => console.error(err)); |

1. **Una vez que tengas el JSON, crea el código en JavaScript para enviar el JSON y obtener la hora de acuerdo con la ubicación del usuario.**

|  |
| --- |
| const options = { method: 'GET' };  fetch('https://ipgeolocation.abstractapi.com/v1/?api\_key=675751b331544912a78c83426eef4445', options)  .then(response => response.json())  .then(response => {  console.log(response);  const latitud = response.latitude, longitud = response.longitude;  const ip = response.ip\_address;  console.log("Tu IP es: ", ip);  console.log("Tus coordenadas son: ", latitud, longitud);  console.log("La hora local es: ", response.timezone.current\_time)  })  .catch(err => console.error(err)); |

**4. Crea una base de datos en MySQL que se llame: Registro.**

CREATE DATABASE registros;

**5. Crea una tabla que se llame: horas, la tabla debe contar con los siguientes campos:**

**a. Longitud**

**b. Latitud**

CREATE TABLE hora (latitud INT, longitude INT);

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Ilustración 10 - Base de datos "registro"

**6. Utiliza Node,js para conectarte con la base de datos.**

a. Crea el código a partir de lo revisado en el tema 8 que te permitirá establecer la

conexión con la base de datos.

|  |
| --- |
| let connection = mysql.createConnection({  host: 'localhost',  // port: '3306',  user: 'root',  password: '',  database: 'registro'  }); |

**7. Almacena en la base de datos la hora obtenida de la API.**

a. Establece la conexión con la base de datos y utiliza el Query Insert Into para almacenar los datos.

|  |
| --- |
| let sql = 'INSERT INTO horas (latitud, longitud) VALUES (?, ?)';  connection.query(sql, [latitud, longitud], function (err, rows) {  if (err) console.log("Error inserting: ", err);  console.log("1 record inserted");  });  connection.end(); |

El código completo es el siguiente:

|  |
| --- |
| let mysql = require('mysql');  const options = { method: 'GET' };  var latitud, longitud, hora;  fetch('https://ipgeolocation.abstractapi.com/v1/?api\_key=675751b331544912a78c83426eef4445', options)  .then(response => response.json())  .then(response => {  console.log(response);  latitud = response.latitude;  longitud = response.longitude;  hora = response.timezone.current\_time;  const ip = response.ip\_address;  console.log("Tu IP es: ", ip);  console.log("Tus coordenadas son: ", latitud, longitud);  console.log("La hora local es: ", hora);  let connection = mysql.createConnection({  host: 'localhost',  // port: '3306',  user: 'root',  password: 'admin2023',  database: 'registro'  });  console.log('Valores a insertar: ', latitud, longitud, hora);  let sql = 'INSERT INTO horas (latitud, longitud, hora) VALUES (?, ?, ?)';  connection.query(sql, [latitud, longitud, hora], function (err, rows) {  if (err) console.log("Error inserting: ", err);  console.log("1 record inserted");  });  connection.end();  })  .catch(err => console.error(err)); |

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteLa siguiente pantalla muestra los registros insertados los cuales incluyen la hora de la consulta a la API

Ilustración 11 - registros insertados incluyendo la hora.

# **Conclusiones**

El desarrollo de back-end, también conocido como desarrollo del lado del servidor, se refiere al desarrollo de la lógica entre bastidores (lo que no ve el usuario) que soporta a las aplicaciones web.

Los back-end se componen de bases de datos, funciones de autenticación/autorización, interfaces de programación de aplicaciones (API) y todos los diferentes componentes que, combinados con un front-end, permiten a los usuarios ver e interactuar con las aplicaciones en su navegador incluyendo el acceso controlado a las bases de datos del servidor.

Entre las actividades que debe realizar un desarrollador de back-end, tenemos las siguientes:

* Escribir código de calidad (limpio y fácil de mantener), incluyendo la creación de prototipos, sistemas completos y la adición de nuevas funciones a los sistemas existentes.
* Diseño y gestión de bases de datos, requiera que el desarrollador de back-end cuente con un amplio conocimiento de las bases de datos relacionales y no relacionales.
* Creación y gestión de APIs como medio de comunicación con el cliente y con otros sistemas.
* Solución de problemas y depuración de aplicaciones.
* Mantenimiento del servidor web.
* Colaboración con desarrolladores front-end.

Los lenguajes de programación más utilizados en el desarrollo back-end son:

* Python
* Java
* C#

Algunas de las bases de datos más populares incluyen:

* MySQL
* Oracle
* Microsoft SQL Server
* PostgreSQL
* MongoDB

Los sistemas operativos más utilizados en servidores son:

* Linux
* Windows Server

Y los servidores Web:

* Apache
* Microsoft-IIS

Por lo que un desarrollador back-end debe ser competente en los lenguajes, manejadores de bases de datos, sistemas operativos y servidores Web mencionados.