

# PROYECTO UNIFICADO DE LAS CARRERAS DE REDES Y TELECOMUNICACIONES Y DESARROLLO DE SOFTWARE



INSTITUTO TECNOLÓGICO SUPERIOR

QUITO METROPOLITANO

FORMANDO PROFESIONALES DE ÉLITE

**ITSQMET**

ITSQMET 2021

EVIDENCIAR EL PROCESO DE  
APRENDIZAJE POR PARTE DE  
LOS ESTUDIANTES

[WWW.ITSQMET.EDU.EC](http://WWW.ITSQMET.EDU.EC)



# **INSTITUTO TECNOLÓGICO SUPERIOR QUITO METROPOLITANO**



INSTITUTO TECNOLÓGICO SUPERIOR  
**QUITO METROPOLITANO**  
FORMANDO PROFESIONALES DE ÉLITE

**ITSQMÉT**

## **PROYECTO DE LA ASIGNATURA CALIDAD DE SOFTWARE**

**ING. DANNY BERMEO OCHOA**

**JOSÉ LUIS FRÍAS HERNÁNDEZ**

**OCTUBRE 2022 – MARZO 2023**

## CONTENIDO

PROYECTO FINAL – “COLLEGE” .....	5
1. INTRODUCCIÓN AL PROYECTO DE APLICACIÓN.....	5
2. OBJETIVO GENERAL DEL PROYECTO.....	5
3. OBJETIVOS ESPECÍFICOS DEL PROYECTO .....	5
4. PLANTEAMIENTO DEL PROBLEMA.....	6
5. JUSTIFICACION .....	6
6. MARCO TEORICO .....	6
6.1. MVC.....	6
6.2. GITHUB.....	8
6.3. HTML VALIDATOR .....	8
7. DESARROLLO Y/O IMPLEMENTACION .....	9
7.1. LEVANTAMIENTO DE LA BASE DE DATOS .....	9
7.2. DIAGRAMA ENTIDAD-RELACION.....	11
7.3. DIAGRAMA DE ARQUITECTURA .....	11
7.4. IMPLEMENTACION DE LA BASE DE DATOS .....	12
7.5. CODIFICACION DE LA PÁGINA .....	14
7.5.1. CÓDIGO HTML EN NETBEANS CON JAVA .....	14
7.5.1.1. INTRODUCCIÓN DEL SITIO WEB .....	14
7.5.1.2. LOGIN DEL SITIO WEB .....	16
7.5.1.3. VISTA DE CARRERAS Y MATERIAS DE LA PÁGINA .....	19
7.5.1.4. VISTA DE ESTUDIANTES DEL SITIO WEB .....	21
7.5.1.5. VISTA DE PROFESORES DEL SITIO WEB.....	22
7.5.2. CONEXIÓN CON LA BASE DE DATOS.....	24
7.5.3. CONTROLADORES .....	24

7.5.3.1.	CONTROLADOR DE CARRERAS .....	24
7.5.3.2.	CONTROLADOR DE LOGIN.....	25
7.5.3.3.	CONTROLADOR PROCEDURES .....	26
7.5.3.4.	CONTROLADOR ESTUDIANTES .....	26
7.5.3.5.	CONTROLADOR DE MATERIAS.....	28
7.5.3.6.	CONTROLADOR DE PROFESORES .....	29
7.5.4.	MODELOS .....	30
7.5.4.1.	CARRERA.....	30
7.5.4.2.	ESTUDIANTES .....	30
7.5.4.3.	MATERIAS .....	31
7.5.4.4.	PROFESORES.....	32
7.5.4.5.	USUARIOS.....	32
7.5.5.	SERVLETS .....	33
7.5.5.1.	SERVLET CARRERAS.....	33
7.5.5.2.	SERVLET LOGIN.....	34
7.5.5.3.	SERVLET PROCEDIMIENTOS .....	35
7.5.5.4.	SERVLET ESTUDIANTES .....	35
7.5.5.5.	SERVLET MATERIAS.....	36
7.5.5.6.	SERVLET PROFESORES .....	37
7.5.6.	DEPENDENCIAS USADAS .....	38
8.	ANÁLISIS DE RESULTADOS.....	38
9.	VERIFICACION DE CALIDAD DE SOFTWARE.....	41
10.	CONCLUSIONES .....	42
11.	RECOMENDACIONES.....	43
12.	BIBLIOGRAFÍA .....	43

## **PROYECTO FINAL – “COLLEGE”**

### **1. INTRODUCCIÓN AL PROYECTO DE APLICACIÓN**

La aplicación “College” es un sistema de registro y control de estudiantes, profesores, información de las asignaturas y carreras además de datos del ámbito educativo, a través de una aplicación web. En la aplicación se podrá visualizar la información, realizar el registro, actualización, así como eliminar tanto a estudiantes como de docentes. Todo esto conectado a una base de datos con procedimientos de almacenamientos, funciones, triggers y también el control de usuarios y roles.

### **2. OBJETIVO GENERAL DEL PROYECTO**

Desarrollada una página web, implementar un patrón de diseño y mediante las herramientas de verificación de software verificar la calidad de software ha sido diseñado.

### **3. OBJETIVOS ESPECÍFICOS DEL PROYECTO**

- Validar que tipo de herramienta de verificación de código es mas idónea para analizar el producto entregado.
- Implementar los conocimientos adquiridos en el presente módulo para tener un producto de calidad.

#### **4. PLANTEAMIENTO DEL PROBLEMA**

En la actualidad empresas e incluso instituciones educativas tienen su propio apartado web, para mostrar y recopilar información, marketing, manejo de datos de sus estudiantes y colaboradores en las diferentes áreas de trabajo.

Una vez realizado el producto final viene la problemática de que calidad de software ha sido desarrollado y en caso de ser necesario poderlo modificar para entregar un producto idóneo

#### **5. JUSTIFICACION**

Si queremos destacarnos en el ámbito profesional debemos realizar buenas prácticas de programación, con una arquitectura, patrones y mas utilidades que nos permitan entregar un producto limpio y de calidad. Por lo cual este presente proyecto nos ayuda a investigar y analizar lo antes mencionado.

#### **6. MARCO TEORICO**

##### **6.1.MVC**

En líneas generales, MVC es una propuesta de arquitectura del software utilizada para separar el código por sus distintas responsabilidades, manteniendo distintas capas que se encargan de hacer una tarea muy concreta, lo que ofrece beneficios diversos.

MVC se usa inicialmente en sistemas donde se requiere el uso de interfaces de usuario, aunque en la práctica el mismo patrón de arquitectura se puede utilizar para distintos tipos de aplicaciones. Surge de la necesidad de crear software más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos.

Su fundamento es la separación del código en tres capas diferentes, acotadas por su responsabilidad, en lo que se llaman Modelos, Vistas y Controladores, o lo que es lo mismo, Model, Views & Controllers, si lo prefieres en inglés.

MVC es un "invento" que ya tiene varias décadas y fue presentado incluso antes de la aparición de la Web. No obstante, en los últimos años ha ganado mucha fuerza y seguidores gracias a la aparición de numerosos frameworks de desarrollo web que utilizan el patrón MVC como modelo para la arquitectura de las aplicaciones web.

- **Modelo:** Es la capa donde se trabaja con los datos, por tanto, contendrá mecanismos para acceder a la información y también para actualizar su estado. Los datos los tendremos habitualmente en una base de datos, por lo que en los modelos tendremos todas las funciones que accederán a las tablas y harán los correspondientes selects, updates, inserts, etc. No obstante, cabe mencionar que cuando se trabaja con MCV lo habitual también es utilizar otras librerías como PDO o algún ORM como Doctrine, que nos permiten trabajar con abstracción de bases de datos y persistencia en objetos. Por ello, en vez de usar directamente sentencias SQL, que suelen depender del motor de base de datos con el que se esté trabajando, se utiliza un dialecto de acceso a datos basado en clases y objetos.
- **Vista:** Las vistas, como su nombre nos hace entender, contienen el código de nuestra aplicación que va a producir la visualización de las interfaces de usuario, o sea, el código que nos permitirá renderizar los estados de nuestra aplicación en HTML. En las vistas nada más tenemos los códigos HTML y PHP que nos permite mostrar la salida. En la vista generalmente trabajamos con los datos, sin embargo, no se realiza un acceso directo a éstos. Las vistas requerirán los datos a los modelos y ellas se generará la salida, tal como nuestra aplicación requiera.
- **Controlador:** Contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como visualizar un elemento, realizar una compra, una búsqueda de información, etc. En realidad, es una capa que sirve de enlace entre las vistas y los modelos, respondiendo a los mecanismos que puedan requerirse para implementar las necesidades de nuestra aplicación.

## **6.2.GITHUB**

GitHub es una de las herramientas más populares para guardar información y compartirla de forma segura, ya sea pública o privada, una especie de nube para programación, según nos indica el documento de GitHub “GitHub es una plataforma de hospedaje de código para el control de versiones y la colaboración. Este permite que tú y otras personas trabajen juntos en proyectos desde donde sea.” (GitHub, 2021)

Fue fundada en el año 2005, para el sistema operativo de Linux, a razón de la necesidad de un DVCS (Distributed Version Control Systems) la cual había ofrecido BitKeeper al sistema operativo, sin embargo pasó a ser de pago, entonces fue cuando Torvalds, fundador de Linux, empezó a desarrollar su propia herramienta de código abierto, siendo una de sus principales objetivos implementar un servicio de soporte, velocidad, procesar proyectos grandes, diseño sencillo, entre otras características. (Chacon & Straub, 2021)

Según un informe de la Universidad de Zaragoza indica que “En abril de 2015, el número de usuarios es más de 9,4 millones y el número de repositorios alcanza los 22,4 millones.” (Lopez Pellicer, Latre, Nogueras, & Zarazaga, 2015) Siendo una herramienta popular entre los desarrolladores ya desde el 2015, aun siendo vigente por su utilidad, radica en su funcionalidad de interactuar con la información almacenada.

## **6.3.HTML VALIDATOR**

Un validador HTML es un programa o aplicación especializada que se utiliza para verificar la validez del marcado HTML en una página web en busca de errores de sintaxis y léxicos. Esto se debe a que HTML no está compilado y es un lenguaje muy indulgente en términos

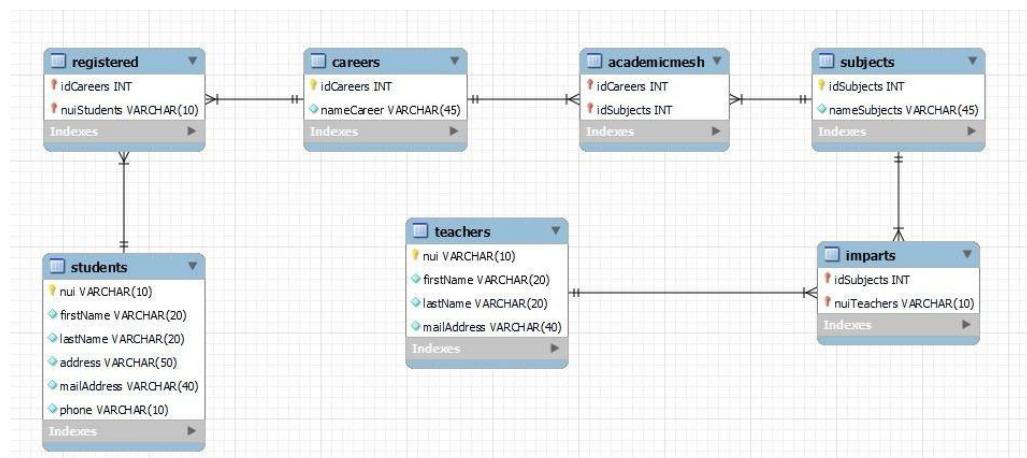
de errores; todo sigue funcionando de alguna manera, por lo que no hay una forma rápida de verificar dichos errores. Por lo tanto, se requiere un programa separado.

Un validador HTML básicamente verifica si los códigos HTML y CSS en una página web cumplen con los estándares establecidos por el Consorcio World Wide Web (W3C). Un ejemplo de un validador de HTML es el propio Validator Suite del W3C que se encuentra en <http://validator.w3.org/>. También hay un validador CSS, llamado Jigsaw, que se encuentra en <http://jigsaw.w3.org/css-validator/>.

## 7. DESARROLLO Y/O IMPLEMENTACION

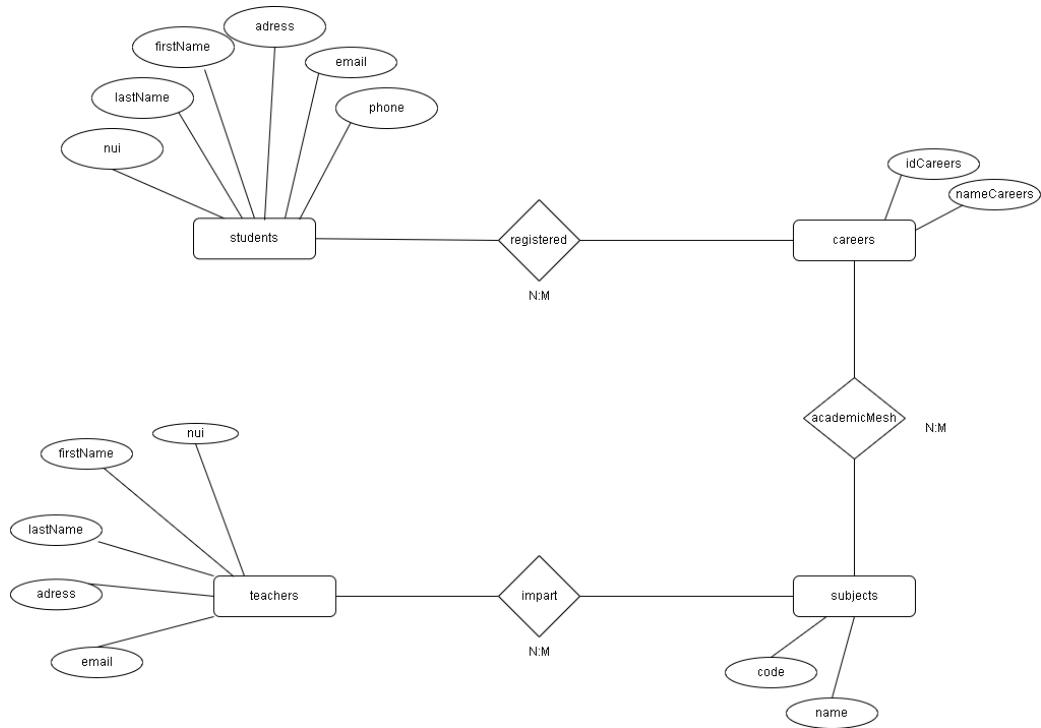
### 7.1. LEVANTAMIENTO DE LA BASE DE DATOS

Para el sistema se va a necesitar una base de datos en la cual se pueda llevar el control de los estudiantes donde se disponga del número de cédula el cual es único e irrepetible, nombres y apellidos, dirección, correo y número de contacto, además será necesario que se disponga de otra tabla donde tengamos el registro para los docentes con su número de cédula que será único e irrepetible, nombres y apellidos, correo y la asignatura que imparte, la parte de asignatura debe relacionarse con los estudiantes y los docentes para cuando se seleccione un docente se carguen los alumnos que están registrados en esa asignatura.

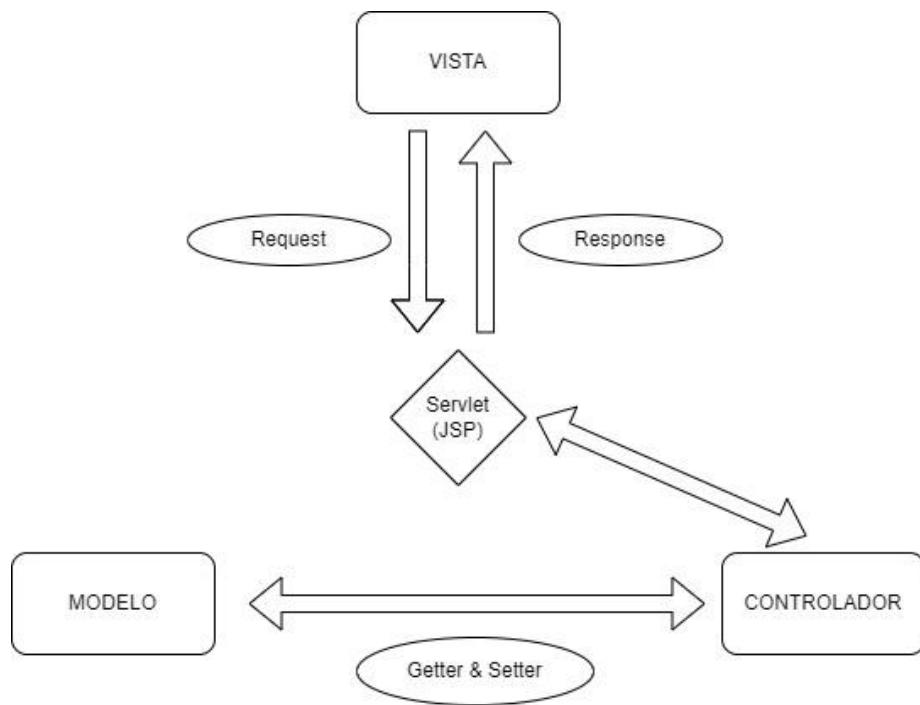


<b>Tablas</b>	<b>Atributos</b>
Students	NUI varchar(10) not null unique primary key  FirstName varchar(20) int not null  LastName varchar(20) int not null  Address varchar (50) int not null  mailAddress varchar(40) int not null  Phone varchar (10) int not null
Teachers	NUI varchar(10) not null unique primary key  FirstName varchar(45) int not null  LastName varchar(45) int not null  mailAddress varchar(45) int not null
Careers	idCareers int not null auto_increment  nameCareer varchar (45) not null
Subjects	idSubjects int not null auto_increment  nameSubjects varchar (45) not null

## 7.2.DIAGRAMA ENTIDAD-RELACION

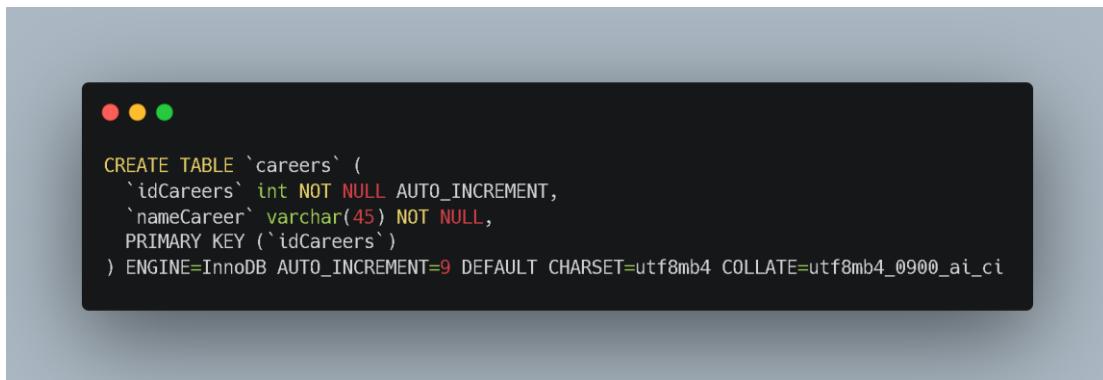


## 7.3.DIAGRAMA DE ARQUITECTURA



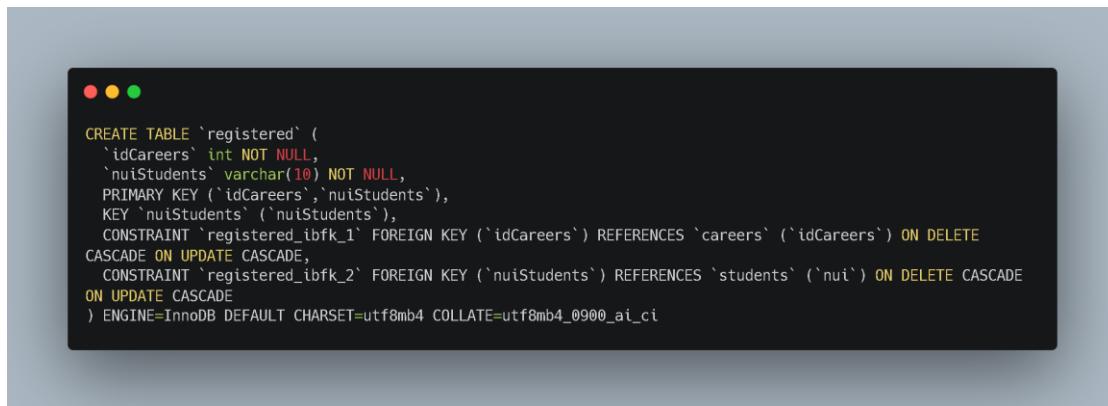
## 7.4.IMPLEMENTACION DE LA BASE DE DATOS

Tabla carrers



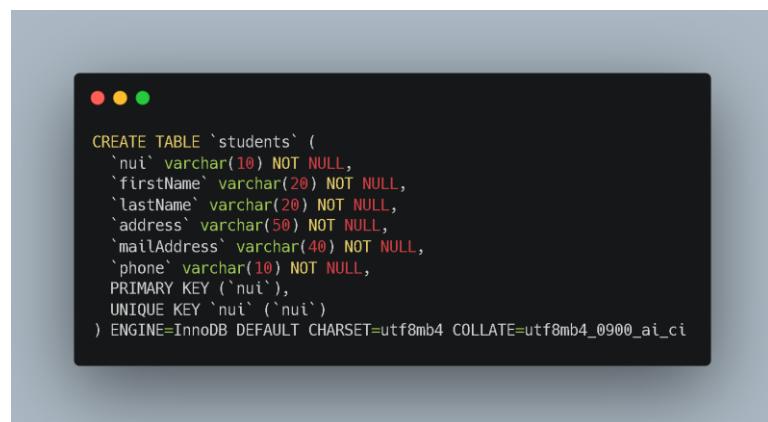
```
CREATE TABLE `careers` (
  `idCareers` int NOT NULL AUTO_INCREMENT,
  `nameCareer` varchar(45) NOT NULL,
  PRIMARY KEY (`idCareers`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Tabla registered



```
CREATE TABLE `registered` (
  `idCareers` int NOT NULL,
  `nuiStudents` varchar(10) NOT NULL,
  PRIMARY KEY (`idCareers`,`nuiStudents`),
  KEY `nuiStudents` (`nuiStudents`),
  CONSTRAINT `registered_ibfk_1` FOREIGN KEY (`idCareers`) REFERENCES `careers` (`idCareers`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `registered_ibfk_2` FOREIGN KEY (`nuiStudents`) REFERENCES `students` (`nui`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Tabla students



```
CREATE TABLE `students` (
  `nui` varchar(10) NOT NULL,
  `firstName` varchar(20) NOT NULL,
  `lastName` varchar(20) NOT NULL,
  `address` varchar(50) NOT NULL,
  `mailAddress` varchar(40) NOT NULL,
  `phone` varchar(10) NOT NULL,
  PRIMARY KEY (`nui`),
  UNIQUE KEY `nui` (`nui`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

## Tabla subjects

```
CREATE TABLE `subjects` (
  `idSubjects` int NOT NULL AUTO_INCREMENT,
  `nameSubjects` varchar(45) NOT NULL,
  PRIMARY KEY (`idSubjects`)
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

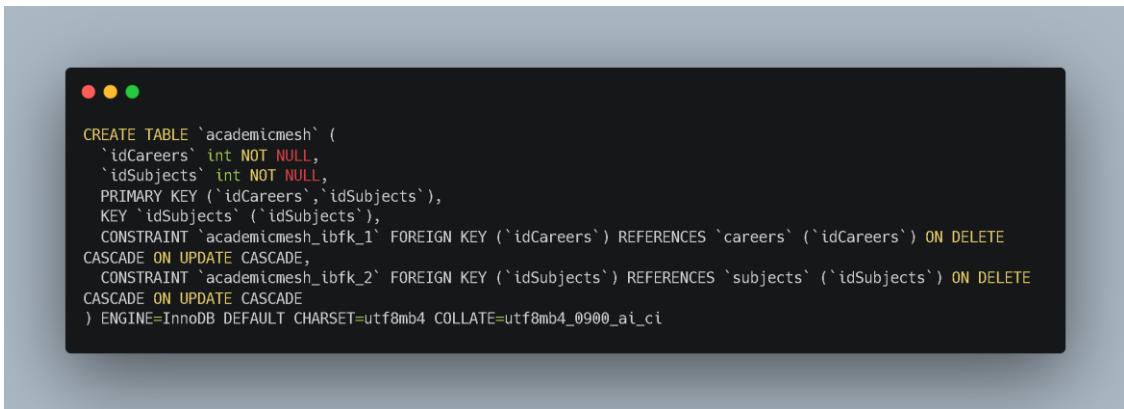
## Tabla teachers

```
CREATE TABLE `teachers` (
  `nui` varchar(10) NOT NULL,
  `firstName` varchar(20) NOT NULL,
  `lastName` varchar(20) NOT NULL,
  `mailAddress` varchar(40) NOT NULL,
  PRIMARY KEY (`nui`),
  UNIQUE KEY `nui` (`nui`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

## Tabla imparts

```
CREATE TABLE `imparts` (
  `idSubjects` int NOT NULL,
  `nuiTeachers` varchar(10) NOT NULL,
  PRIMARY KEY (`idSubjects`,`nuiTeachers`),
  KEY `nuiTeachers`(`nuiTeachers`),
  CONSTRAINT `imparts_ibfk_1` FOREIGN KEY (`idSubjects`) REFERENCES `subjects`(`idSubjects`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `imparts_ibfk_2` FOREIGN KEY (`nuiTeachers`) REFERENCES `teachers`(`nui`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

## Tabla academicmech

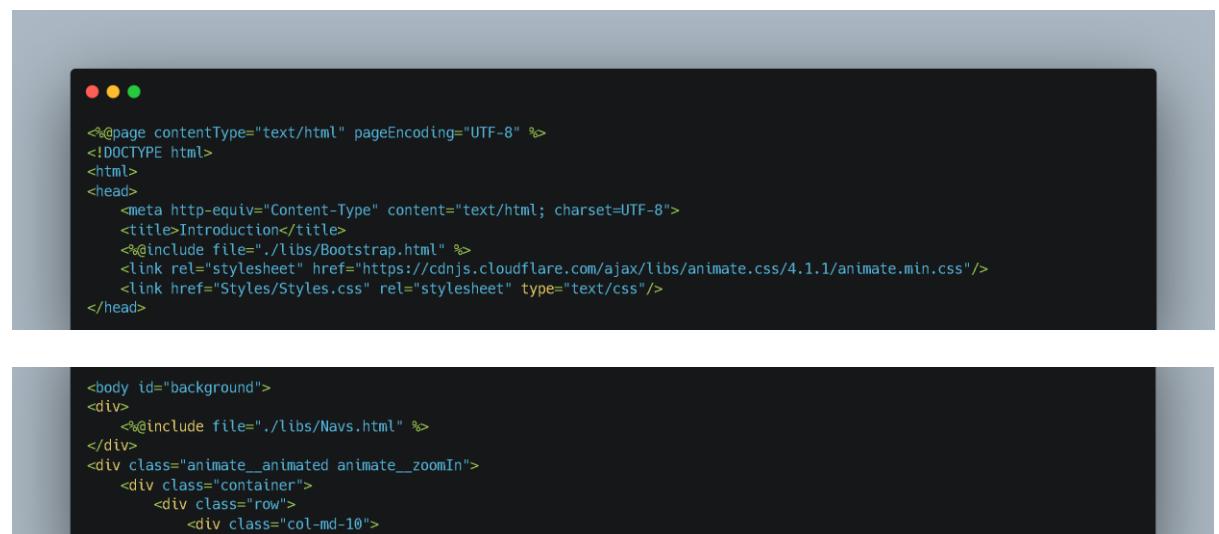


```
CREATE TABLE `academicmesh` (
  `idCareers` int NOT NULL,
  `idSubjects` int NOT NULL,
  PRIMARY KEY (`idCareers`,`idSubjects`),
  KEY `idSubjects` (`idSubjects`),
  CONSTRAINT `academicmesh_ibfk_1` FOREIGN KEY (`idCareers`) REFERENCES `careers` (`idCareers`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `academicmesh_ibfk_2` FOREIGN KEY (`idSubjects`) REFERENCES `subjects` (`idSubjects`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

## 7.5.CODIFICACION DE LA PÁGINA

### 7.5.1. CÓDIGO HTML EN NETBEANS CON JAVA

#### 7.5.1.1.INTRODUCCIÓN DEL SITIO WEB



```
<%@page contentType="text/html" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Introduction</title>
    <%@include file="./libs/Bootstrap.html" %>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css"/>
    <link href="Styles/Styles.css" rel="stylesheet" type="text/css"/>
</head>

<body id="background">
<div>
    <%@include file="./libs/Navs.html" %>
</div>
<div class="animate__animated animate__zoomIn">
    <div class="container">
        <div class="row">
            <div class="col-md-10">
```

```
<br>
<br>
<br>
<br>
<br>
<br>
<center>
    <h1 id="title">Introducción</h1>
</center>
<br>
<p align="justify">
    <b>
        La aplicación "College" es un sistema de registro y control de estudiantes, profesores, información de las asignaturas y carreras además de datos del ámbito educativo, a través de una aplicación web. En la aplicación se podrá visualizar la información, realizar el registro, actualización así como eliminar tanto a estudiantes como de docentes. Todo esto conectado a una base de datos con procedimientos de almacenamientos, funciones, triggers y también el control de usuarios y roles.
    </b>
</p>
</div>
```

```
<br>
<br>
<br>
<br>
</div>
<div class="container">
    <div class="row">
        <div class="col-md-6">
            <table>
                <tr>
                    <td>
                        <h1 id="title">Objetivo General</h1>
                        <p align="justify" style="color: #141c27"><b>
                            • Desarrollar una aplicación WEB con el conocimiento adquirido de programación y base de datos usando NETBEANS y MYSQL, tanto en la parte de personalización del programa y el desarrollo de código con el lenguaje de JAVA, mediante un enfoque teórico/práctico, además de utilizar lenguajes como HTML, JS y CSS para poder realizar la vista del proyecto</b>
                        </p>
                    </td>
                </tr>
            </table>
        </div>
    </div>
</div>
```

```
</td>
</tr>
</table>
</div>
<div class="col-md-6">
    <table>
        <tr>
            <td>
                <h1 id="title">Objetivos Específicos</h1>
                <p align="justify" style="color: #141c27"><b>
                    • Describir las características y funciones que nos brindan los programas usados en la práctica.<br>
                    • Conocer y explicar la ejecución del programa, su funcionamiento dentro de la web y su respectivo lenguaje de código usado en la programación
                    <br>• Implementar una base datos, con los respectivos pasos explicados en clases, para una correcta organización del programa al guardar los datos.</b>
                </p>
            </td>
        </tr>
    </table>
</div>
```

```
</td>
</tr>
</table>
</div>
</div>
<br>
<br>
</div>
</div>
```

```

<footer>
    <div>
        <div class="container">
            <div class="row">
                <div class="col-md-6">
                    <label>INSTITUTO TÉCNICO SUPERIOR QUITO METROPOLITANO</label>
                </div>
                <div class="col-md-6" style="text-align: left">
                    <label>
                        Desarrollado por: Joao Jaramillo, José Luis Frías y Daniel Parrales
                    </label>
                </div>
            </div>
        </div>
    </div>
</body>
</html>

```

### 7.5.1.2.LOGIN DEL SITIO WEB

```

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <%@include file="./libs/Bootstrap.html" %>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css"/>
    <link href="Styles/Styles.css" rel="stylesheet" type="text/css"/>
    <title>Login</title>
</head>
<body id="background">
<header>
    <h1 class="text-center" style="padding: 6rem 0">
        
        <br>
        Bienvenido a COLLEGE</h1>
</header>

<div class="container-fluid">
    <section class="w-100 p-4 d-flex justify-content-center pb-4">
        <div style="width: 26rem" id="loginAndRegister">
            <!-- Pills navs -->
            <nav class="nav nav-pills nav-justified">
                <button
                    class="nav-link active primary"
                    id="pills-home-tab"
                    data-bs-toggle="pill"
                    data-bs-target="#pills-login"
                    type="button"
                    role="tab"
                    aria-controls="pills-login"
                    aria-selected="true"
                >
                    Iniciar Sesión
                </button>
                <button
                    class="nav-link"
                    id="pills-profile-tab"
                    data-bs-toggle="pill"
                    data-bs-target="#pills-register"
                    type="button"
                    role="tab"
                    aria-controls="pills-register"
                    aria-selected="false"
                >
                    Registrarse
                </button>
            </nav>
        </div>
    </section>
</div>

```

```
<!-- Pills navs -->  
<!-- Pills content -->  
<div class="tab-content">  
    <div  
        class="tab-pane fade active show"  
        id="pills-login"  
        role="tabpanel"  
        aria-labelledby="tab-login"  
    >
```

```
<form method="get" action="login">  
    <p class="text-center mt-3">Por favor, ingrese a su cuenta</p>  
  
    <!-- Email input -->  
    <div class="form-floating mb-3">  
        <input  
            type="email"  
            class="form-control"  
            name="mail"  
            id="loginEmail"  
            placeholder="name@example.com"  
        />
```

```
        <label for="loginEmail">Correo Electrónico</label>  
    </div>  
  
    <!-- Password input -->  
    <div class="form-floating mb-3">  
        <input  
            type="password"  
            class="form-control"  
            name="pass"  
            id="loginPassword"  
            placeholder="Password"  
        />  
        <label for="loginPassword">Contraseña</label>  
    </div>
```

```
<!-- Submit button -->  
    <button type="submit" class="btn btn-primary btn-block mb-3">  
        Ingresar  
    </button>  
    </form>  
    </div>  
    <div  
        class="tab-pane fade"  
        id="pills-register"  
        role="tabpanel"  
        aria-labelledby="tab-register"  
    >
```

```
<form action="login" method="post">  
    <p class="text-center">Por favor, ingrese a sus datos</p>  
    <!-- Email input -->  
    <div class="form-floating mb-3">  
        <input  
            type="email"  
            class="form-control"  
            id="registerEmail"  
            placeholder="name@example.com"  
            name="mail"  
        />  
        <label for="registerEmail">Email address</label>  
    </div>
```

```
    <!-- Password input -->  
    <div class="form-floating mb-3">  
        <input  
            type="password"  
            class="form-control"  
            id="registerPassword"  
            placeholder="Password"  
            name="pass"  
        />  
        <label for="registerPassword">Password</label>  
    </div>
```

```

        <!-- Submit button -->
        <button type="submit" class="btn btn-primary btn-block mb-3">
            Registrarse
        </button>
    </form>
</div>
</div>

```

```

        <!-- Pills content -->
        </div>
    </section>
</div>
<footer>
<div>
    <div class="container">
        <div class="row">
            <div class="col-md-6">
                <label>INSTITUTO TÉCNICO SUPERIOR QUITO METROPOLITANO</label>
            </div>
            <div class="col-md-6" style="text-align: left">
                <label>
                    Desarrollado por: Joao Jaramillo, José Luis Frías y Daniel Parrales
                </label>
            </div>
        </div>
    </div>
</footer>
</body>
</html>

```

```

<%@page contentType="text/html" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css"/>
    <%@include file="./libs/Bootstrap.html" %>
    <%@include file="./libs/DialogMessages.html" %>
    <link href="Styles/Styles.css" rel="stylesheet" type="text/css"/>
    <script src="js/procedures.js"></script>
    <title>Procedimientos Almacenados</title>
</head>

```

```

<body id="background">
<div>
    <%@include file="./libs/Mnavs.html" %>
</div>
<br>
<br>
<br>
<div class="container text-center">
    <h1 class="mb-4 animate__animated animate__bounce">Validar tipo de cuenta de correo</h1>
    <div class="row justify-content-center">
        <div class="col-4">
            <form action="procedures/correoEst" id="correoEst" onsubmit="callProcedure(event, this)">

```

```

                <div class="form-floating mb-3">
                    <input type="text" class="form-control"
                        name="nui"
                        id="insertNui"
                        placeholder="Ingrese el NUI"/>
                    <label for="insertNui">Ingrese el NUI</label>
                </div>
            </form>
        </div>
    
```

```

<div class="col-2">
    <button class="button" form="correoEst">
        <span class="button-content"
            style="color: white">
            Validar
        </span>
    </button>
</div>
<div>
    <p id="response"></p>
</div>
</div>
</body>
</html>

```

### 7.5.1.3.VISTA DE CARRERAS Y MATERIAS DE LA PÁGINA



```

<@page contentType="text/html" pageEncoding="UTF-8">
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Carreras y Materias</title>
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css"/>
        <%@include file=".libs/Bootstrap.html" %>
        <%@include file=".libs/DialogMessages.html" %>
        <link href="Styles/Styles.css" rel="stylesheet" type="text/css"/>
        <script src="js/Methods.js"></script>
    </head>

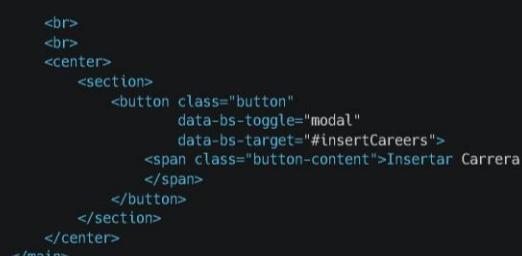
```



```

<body id="background">
    <div>
        <%@include file=".libs/Navs.html" %>
    </div>
    <div class="container">
        <div class="row">
            <div class="col-md-6">
                <main>
                    <br>
                    <br>
                    <header>
                        <h1 align="center" style="font-family: Roboto; color: #141c27" class="animate__animated animate__bounce">
                            Carreras</h1>
                    </header>

```



```

                    <br>
                    <br>
                    <center>
                        <section>
                            <button class="button"
                                data-bs-toggle="modal"
                                data-bs-target="#insertCareers">
                                <span class="button-content">Insertar Carrera
                            </span>
                        </button>
                    </section>
                </center>
            </main>

```

```
<div class="container" align="center" style="font-family: Roboto">
  <table
    id="table"
    data-locale="es-ES"
    data-url="/college/careers"
    data-toggle="table"
    data-filter-control="true"
    data-height = "500"
    data-show-columns="true"
    data-search="true"
    data-show-export="true">
    <thead>
```

```
<tr>
    <th data-field="idCareers"><center>ID</center></th>
    <th data-field="nameCareer"><center>Carrera</center></th>
    <th data-field="operate" data-formatter="operations"
        data-width="222"><center>Acción</center></th>
</tr>
</thead>
</table>
</div>
</div>
<div class="col-md-6">
<main>
    <br>
    <br>
    <header>
        <h1 align="center" style="font-family: Roboto,serif; color: #141c27">
            <span class="animate__animated animate__bounce">
                Asignaturas</h1>
    </header>
```

<center>ID</center>	<center>Asignatura</center>	<center>Acción</center>

#### 7.5.1.4. VISTA DE ESTUDIANTES DEL SITIO WEB

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Estudiantes</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css"/>
    <%@include file=".libs/Bootstrap.html" %>
    <%@include file=".libs/DialogMessages.html" %>
    <link href="Styles/Styles.css" rel="stylesheet" type="text/css"/>
    <script src="js/Methods.js"></script>
</head>

<body id="background">
<div>
    <%@include file=".libs/Navs.html" %>
</div>
<div>
    <main>
        <br>
        <br>
        <header>
            <h1 align="center" style="font-family: Roboto; color: #141c27" class="animate__animated animate__bounce">
                Control para Estudiantes</h1>
        </header>
        <br>
        <br>
        <center>
            <section>
                <button class="button"
                    data-bs-toggle="modal"
                    data-bs-target="#insertStudents">
                    <span class="button-content">Insertar nuevo
                    Estudiante </span>
                </button>
            </section>
        </center>
    </main>
</div>

<br>
<div class="container" align="center" style="font-family: Roboto">
    <table
        id="table"
        data-locale="es-ES"
        data-url="/college/students"
        data-toggle="table"
        data-filter-control="true"
        data-height="500"
        data-show-columns="true"
        data-search="true"
        data-show-export="true">
        <thead>

<br>
<div class="container" align="center" style="font-family: Roboto">
    <table
        id="table"
        data-locale="es-ES"
        data-url="/college/students"
        data-toggle="table"
        data-filter-control="true"
        data-height="500"
        data-show-columns="true"
        data-search="true"
        data-show-export="true">
        <thead>
```

```

<tr>
    <th data-field="nui">
        <center>NUI</center>
    </th>
    <th data-field="firstName">
        <center>Nombre</center>
    </th>
    <th data-field="lastName">
        <center>Apellido</center>
    </th>
    <th data-field="address">
        <center>Dirección</center>
    </th>
    <th data-field="mailAddress">
        <center>Correo</center>
    </th>
    <th data-field="phone">
        <center>Teléfono</center>
    </th>
    <th data-field="operate" data-formatter="operations"
        data-width="222">
        <center>Acción</center>
    </th>
</tr>
</thead>

```

```

        </table>
    </div>
</div>
<footer>
<div>
    <div class="container">
        <div class="row">
            <div class="col-md-6">
                <label>INSTITUTO TÉCNICO SUPERIOR QUITO METROPOLITANO</label>
            </div>
            <div class="col-md-6" style="text-align: left">
                <label>
                    Desarrollado por: Joao Jaramillo, José Luis Frías y Daniel Parrales
                </label>
            </div>
        </div>
    </div>
</footer>
<%@include file="./libs/ModalsStudents.html" %>
</body>
</html>

```

### 7.5.1.5.VISTA DE PROFESORES DEL SITIO WEB

```

● ● ●
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Profesores</title>
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css"/>
        <%@include file="./libs/Bootstrap.html" %>
        <%@include file="./libs/DialogMessages.html" %>
        <link href="Styles/Styles.css" rel="stylesheet" type="text/css"/>
        <script src="js/Methods.js"></script>
    </head>

```

```

<body id="background">
    <div>
        <%@include file="./libs/Navs.html"%>
    </div>
    <div>
        <main>
            <br>
            <br>
            <header>
                <h1 align="center" style="font-family: Roboto; color: #141c27" class="animate__animated animate__bounce">
                    Control para Docentes</h1>
            </header>
            <br>
            <br>

```

```

        <center>
            <section>
                <button class="button"
                    data-bs-toggle="modal"
                    data-bs-target="#insertTeachers">
                    <span class="button-content">Insertar nuevo
                    Docente </span>
                </button>
            </section>
        </center>
    </main>
    <br>

```

```

    <div class="container" align="center" style="font-family: Roboto">
        <table
            id="table"
            data-locale="es-ES"
            data-url="/college/teachers"
            data-toggle="table"
            data-filter-control="true"
            data-height = "500"
            data-show-columns="true"
            data-search="true"
            data-show-export="true">
            <thead>

```

```

                <tr>
                    <th data-field="nui"><center>NUI</center></th>
                    <th data-field="firstName"><center>Nombre</center></th>
                    <th data-field="lastName"><center>Apellido</center></th>
                    <th data-field="mailAddress"><center>Correo</center></th>
                    <th data-field="operate" data-formatter="operations"
                        data-width="222"><center>Acción</center></th>
                </tr>
            </thead>
        </table>
    </div>

```

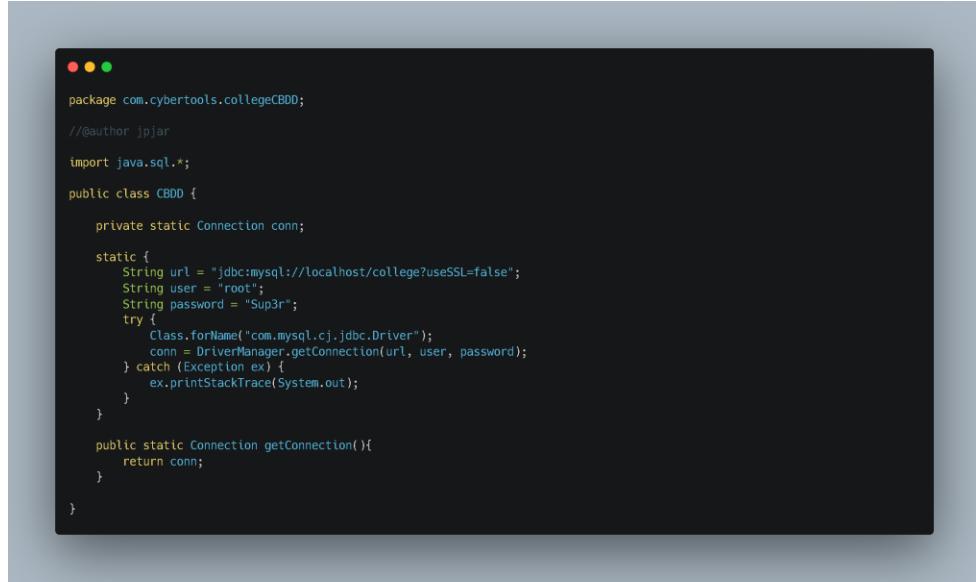
  

```

    <footer>
        <div>
            <div class="container">
                <div class="row">
                    <div class="col-md-6">
                        <label>INSTITUTO TÉCNICO SUPERIOR QUITO METROPOLITANO</label>
                    </div>
                    <div class="col-md-6" style="text-align: left">
                        <label>
                            Desarrollado por: Joao Jaramillo, José Luis Frías y Daniel Parrales
                        </label>
                    </div>
                </div>
            </div>
        </div>
        <%@include file="./libs/ModalsTeachers.html"%>
    </body>
</html>

```

## 7.5.2. CONEXIÓN CON LA BASE DE DATOS



```
package com.cybertools.collegeCBDD;

//@author jjar
import java.sql.*;
public class CBDD {

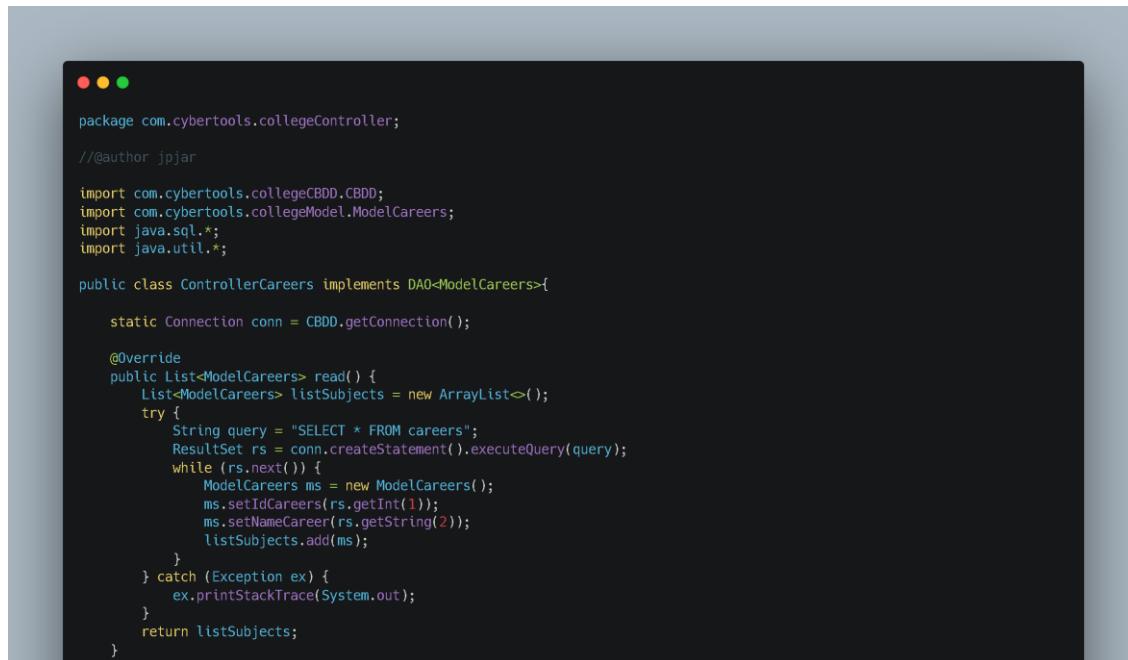
    private static Connection conn;

    static {
        String url = "jdbc:mysql://localhost/college?useSSL=false";
        String user = "root";
        String password = "Sup3r";
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            conn = DriverManager.getConnection(url, user, password);
        } catch (Exception ex) {
            ex.printStackTrace(System.out);
        }
    }

    public static Connection getConnection(){
        return conn;
    }
}
```

## 7.5.3. CONTROLADORES

### 7.5.3.1. CONTROLADOR DE CARRERAS



```
package com.cybertools.collegeController;

//@author jjar
import com.cybertools.collegeCBDD.CBDD;
import com.cybertools.collegeModel.ModelCareers;
import java.sql.*;
import java.util.*;

public class ControllerCareers implements DAO<ModelCareers>{

    static Connection conn = CBDD.getConnection();

    @Override
    public List<ModelCareers> read() {
        List<ModelCareers> listSubjects = new ArrayList<>();
        try {
            String query = "SELECT * FROM careers";
            ResultSet rs = conn.createStatement().executeQuery(query);
            while (rs.next()) {
                ModelCareers ms = new ModelCareers();
                ms.setIdCareers(rs.getInt(1));
                ms.setNameCareer(rs.getString(2));
                listSubjects.add(ms);
            }
        } catch (Exception ex) {
            ex.printStackTrace(System.out);
        }
        return listSubjects;
    }
}
```

```

@Override
public boolean create(ModelCareers t) {
    try {
        String query = "INSERT INTO careers (nameCareer) VALUES (?)";
        PreparedStatement ps = conn.prepareStatement(query);
        ps.setString(1, t.getNameCareer());
        return ps.executeUpdate() != 0; //si no se ejecuta
    } catch (Exception ex) {
        ex.printStackTrace(System.out);
    }
    return false;
}

@Override
public boolean update(ModelCareers t) {
    try {
        String query = "UPDATE careers SET nameCareer=? WHERE idCareers=?";
        PreparedStatement ps = conn.prepareStatement(query);
        ps.setInt(1, t.getIdCareers());
        ps.setString(1, t.getNameCareer());
        return ps.executeUpdate() != 0;
    } catch (Exception ex) {
        ex.printStackTrace(System.out);
    }
    return false;
}

@Override
public boolean delete(ModelCareers t) {
    try {
        String query = "DELETE FROM careers WHERE idCareers=?";
        PreparedStatement ps = conn.prepareStatement(query);
        ps.setInt(1, t.getIdCareers());
        return ps.executeUpdate() != 0;
    } catch (Exception ex) {
        ex.printStackTrace(System.out);
    }
    return false;
}
}

```

### 7.5.3.2. CONTROLADOR DE LOGIN

```

package com.cybertools.collegeController;

import com.cybertools.collegeModel.ModelUsers;
import com.cybertools.collegeCBDD.CBDD;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class ControllerLogin implements DAOLogin<ModelUsers> {

    static Connection conn = CBDD.getConnection();

    @Override
    public boolean login(ModelUsers modelUsers) {
        boolean result = false;
        try {
            String query = "SELECT * FROM usersLogin WHERE mail=? AND pass=md5(?)";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setString(1, modelUsers.getMail());
            ps.setString(2, modelUsers.getPass());
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                result = true;
            }
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
        return result;
    }
}

```

```
    @Override
    public boolean register(ModelUsers modelUsers) {
        try {
            String query = "INSERT INTO usersLogin (mail,pass) VALUES (?,md5(?))";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setString(1, modelUsers.getMail());
            ps.setString(2, modelUsers.getPass());
            return ps.executeUpdate() != 0;
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }
}
```

### 7.5.3.3. CONTROLADOR PROCEDURES

```
● ● ●

package com.cybertools.collegeController;

import com.cybertools.collegeCBDD.CBDD;
import com.cybertools.collegeModel.ModelStudents;
import java.sql.*;

public class ControllerProcedures implements DAOProcedures<ModelStudents>{
    static Connection conn = CBDD.getConnection();

    @Override
    public String correoEst(ModelStudents modelStudents){
        String result = null;
        try {
            CallableStatement cs = conn.prepareCall("{call correoEst(?)}");
            cs.setString(1,modelStudents.getNull());
            ResultSet rs = cs.executeQuery();
            while(rs.next()){
                result = rs.getString(1);
            }
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
        return result;
    }
}
```

### 7.5.3.4. CONTROLADOR ESTUDIANTES

```
● ● ●

package com.cybertools.collegeController;
//@author jpjar

import com.cybertools.collegeCBDD.CBDD;
import com.cybertools.collegeModel.ModelStudents;
import java.sql.*;
import java.util.*;

public class ControllerStudents implements DAO <ModelStudents>{

    static Connection conn = CBDD.getConnection();
```

```

@Override
public List<ModelStudents> read() {
    List<ModelStudents> listStudents = new ArrayList<>();
    try {
        String query = "SELECT * FROM students";
        ResultSet rs = conn.createStatement().executeQuery(query);
        while (rs.next()){
            ModelStudents ms = new ModelStudents();
            ms.setNui(rs.getString(1));
            ms.setFirstName(rs.getString(2));
            ms.setLastName(rs.getString(3));
            ms.setAddress(rs.getString(4));
            ms.setMailAddress(rs.getString(5));
            ms.setPhone(rs.getString(6));
            listStudents.add(ms);
        }
    }catch (Exception ex) {
        ex.printStackTrace(System.out);
    }
    return listStudents;
}

@Override
public boolean create(ModelStudents t) {
    try {
        String query = "INSERT INTO students (nui, firstName, lastName, address, mailAddress, phone) VALUES
        (?, ?, ?, ?, ?, ?)";
        PreparedStatement ps = conn.prepareStatement(query);
        ps.setString(1, t.getNui());
        ps.setString(2, t.getFirstName());
        ps.setString(3, t.getLastName());
        ps.setString(4, t.getAddress());
        ps.setString(5, t.getMailAddress());
        ps.setString(6, t.getPhone());
        return ps.executeUpdate() != 0; //si no se ejecuta
    }catch (Exception ex) {
        ex.printStackTrace(System.out);
    }
    return false; //retorna el falso
}

@Override
public boolean update(ModelStudents t) {
    try {
        String query = "UPDATE students SET firstName=?, lastName=?, address=?, mailAddress=?, phone=? WHERE nui=?";
        PreparedStatement ps = conn.prepareStatement(query);
        ps.setString(6, t.getNui());
        ps.setString(1, t.getFirstName());
        ps.setString(2, t.getLastName());
        ps.setString(3, t.getAddress());
        ps.setString(4, t.getMailAddress());
        ps.setString(5, t.getPhone());
        return ps.executeUpdate() != 0;
    }catch (Exception ex) {
        ex.printStackTrace(System.out);
    }
    return false;
}

@Override
public boolean delete(ModelStudents t) {
    try {
        String query = "DELETE FROM students WHERE nui=?";
        PreparedStatement ps = conn.prepareStatement(query);
        ps.setString(1, t.getNui());
        System.out.println(t);
        System.out.println(ps.toString());
        return ps.executeUpdate() != 0;
    }catch (Exception ex) {
        ex.printStackTrace(System.out);
    }
    return false;
}
}

```

### 7.5.3.5. CONTROLADOR DE MATERIAS

```
package com.cybertools.collegeController;

//@author jpjar

import com.cybertools.collegeCBDD.*;
import com.cybertools.collegeModel.ModelSubjects;
import java.sql.*;
import java.util.*;

public class ControllerSubjects implements DAO<ModelSubjects>{

    static Connection conn = CBDD.getConnection();

    @Override
    public List<ModelSubjects> read() {
        List<ModelSubjects> listSubjects = new ArrayList<>();
        try {
            String query = "SELECT * FROM subjects";
            ResultSet rs = conn.createStatement().executeQuery(query);
            while (rs.next()){
                ModelSubjects ms = new ModelSubjects();
                ms.setIdSubjects(rs.getInt(1));
                ms.setNameSubjects(rs.getString(2));
                listSubjects.add(ms);
            }
        }catch (Exception ex) {
            ex.printStackTrace(System.out);
        }
        return listSubjects;
    }

    @Override
    public boolean create(ModelSubjects t) {
        try {
            String query = "INSERT INTO subjects (nameSubjects) VALUES (?)";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setString(1, t.getNameSubjects());
            return ps.executeUpdate() != 0; //si no se ejecuta
        }catch (Exception ex) {
            ex.printStackTrace(System.out);
        }
        return false; //retorna el falso
    }

    @Override
    public boolean update(ModelSubjects t) {
        try {
            String query = "UPDATE subjects SET nameSubjects=? WHERE idSubjects=?";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setInt(2, t.getIdSubjects());
            ps.setString(1, t.getNameSubjects());
            return ps.executeUpdate() != 0;
        }catch (Exception ex) {
            ex.printStackTrace(System.out);
        }
        return false;
    }

    @Override
    public boolean delete(ModelSubjects t) {
        try {
            String query = "DELETE FROM subjects WHERE idSubjects=?";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setInt(1, t.getIdSubjects());
            System.out.println(t);
            System.out.println(ps.toString());
            return ps.executeUpdate() != 0;
        }catch (Exception ex) {
            ex.printStackTrace(System.out);
        }
        return false;
    }
}
```

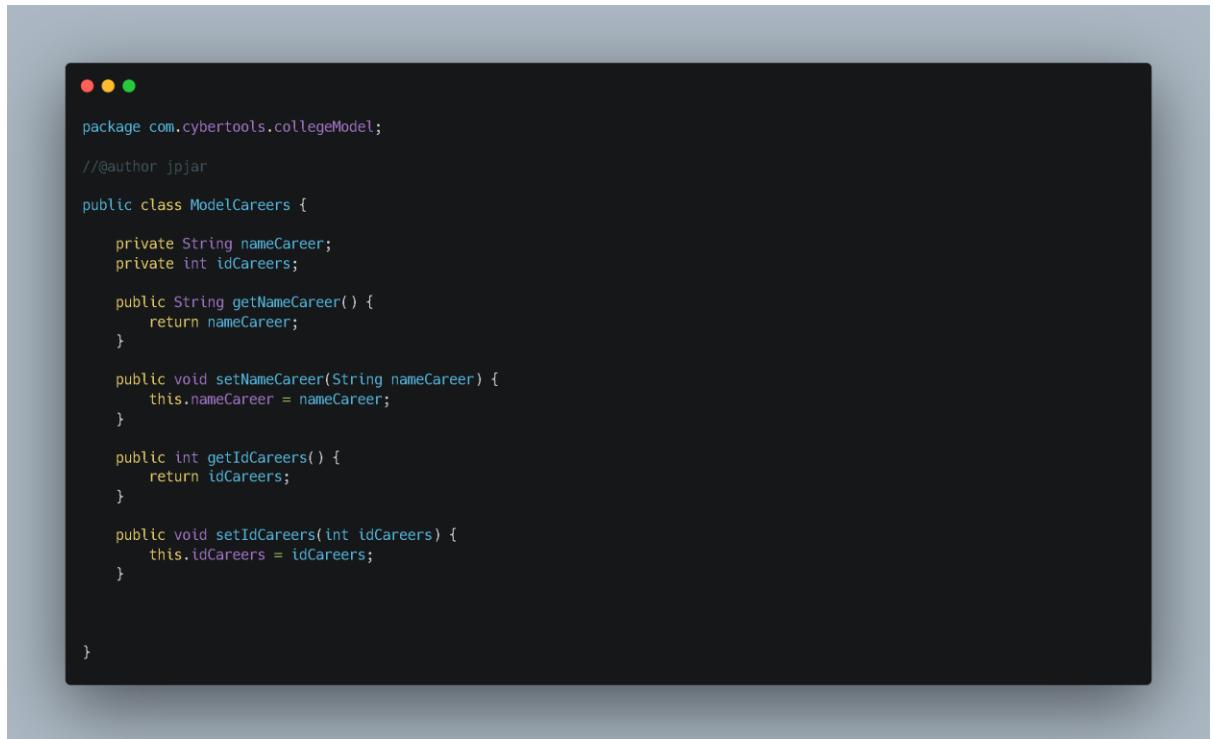
### 7.5.3.6. CONTROLADOR DE PROFESORES

```
package com.cybertools.collegeController;  
//@author jpjar  
  
import com.cybertools.collegeCBDD.CBDD;  
import com.cybertools.collegeModel.ModelTeachers;  
import java.sql.*;  
import java.util.*;  
  
public class ControllerTeachers implements DAO <ModelTeachers>{  
  
    static Connection conn = CBDD.getConnection();  
  
    @Override  
    public List<ModelTeachers> read() {  
        List<ModelTeachers> listTeachers = new ArrayList<>();  
        try {  
            String query = "SELECT * FROM teachers";  
            ResultSet rs = conn.createStatement().executeQuery(query);  
            while (rs.next()) {  
                ModelTeachers mt = new ModelTeachers();  
                mt.setNui(rs.getString(1));  
                mt.setFirstName(rs.getString(2));  
                mt.setLastName(rs.getString(3));  
                mt.setMailAddress(rs.getString(4));  
                listTeachers.add(mt);  
            }  
        } catch (Exception ex) {  
            ex.printStackTrace(System.out);  
        }  
        return listTeachers;  
    }  
  
    @Override  
    public boolean create(ModelTeachers t) {  
        try {  
            String query = "INSERT INTO teachers (nui, firstName, lastName, mailAddress) VALUES (?, ?, ?, ?)";  
            PreparedStatement ps = conn.prepareStatement(query);  
            ps.setString(1, t.getNui());  
            ps.setString(2, t.getFirstName());  
            ps.setString(3, t.getLastName());  
            ps.setString(4, t.getMailAddress());  
            return ps.executeUpdate() != 0; //si no se ejecuta  
        } catch (Exception ex) {  
            ex.printStackTrace(System.out);  
        }  
        return false;  
    }  
  
    @Override  
    public boolean update(ModelTeachers t) {  
        try {  
            String query = "UPDATE teachers SET firstName=?, lastName=?, mailAddress=? WHERE nui=?";  
            PreparedStatement ps = conn.prepareStatement(query);  
            ps.setString(4, t.getNui());  
            ps.setString(1, t.getFirstName());  
            ps.setString(2, t.getLastName());  
            ps.setString(3, t.getMailAddress());  
            return ps.executeUpdate() != 0;  
        } catch (Exception ex) {  
            ex.printStackTrace(System.out);  
        }  
        return false;  
    }  
}
```

```
@Override  
public boolean delete(ModelTeachers t) {  
    try {  
        String query = "DELETE FROM teachers WHERE nui=?";  
        PreparedStatement ps = conn.prepareStatement(query);  
        ps.setString(1, t.getNui());  
        return ps.executeUpdate() != 0;  
    } catch (Exception ex) {  
        ex.printStackTrace(System.out);  
    }  
    return false;  
}
```

## 7.5.4. MODELOS

### 7.5.4.1.CARRERA



```
package com.cybertools.collegeModel;
//@author jpjar
public class ModelCareers {
    private String nameCareer;
    private int idCareers;
    public String getNameCareer() {
        return nameCareer;
    }
    public void setNameCareer(String nameCareer) {
        this.nameCareer = nameCareer;
    }
    public int getIdCareers() {
        return idCareers;
    }
    public void setIdCareers(int idCareers) {
        this.idCareers = idCareers;
    }
}
```

### 7.5.4.2. ESTUDIANTES



```
package com.cybertools.collegeModel;
//@author jpjar
public class ModelStudents {
    private String nui, firstName, lastName, address, mailAddress, phone;
    public String getNui() {
        return nui;
    }
}
```

```
public void setNui(String nui) {
    this.nui = nui;
}

public String getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public String getAddress() {
    return address;
}

public void setAddress(String address) {
    this.address = address;
}

public String getMailAddress() {
    return mailAddress;
}

public void setMailAddress(String mailAddress) {
    this.mailAddress = mailAddress;
}

public String getPhone() {
    return phone;
}

public void setPhone(String phone) {
    this.phone = phone;
}

}
```

#### 7.5.4.3.MATERIAS

```
package com.cybertools.collegeModel;

//@author jpjar

public class ModelSubjects {

    private int idSubjects;
    private String nameSubjects;

    public int getIdSubjects() {
        return idSubjects;
    }
}
```

```
public void setIdSubjects(int idSubjects) {
    this.idSubjects = idSubjects;
}

public String getNameSubjects() {
    return nameSubjects;
}

public void setNameSubjects(String nameSubjects) {
    this.nameSubjects = nameSubjects;
}
}
```

#### 7.5.4.4.PROFESORES

```
package com.cybertools.collegeModel;

//@author jjar
public class ModelTeachers {

    private String nui, firstName, lastName, mailAddress;

    public String getNui() {
        return nui;
    }

    public void setNui(String nui) {
        this.nui = nui;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getMailAddress() {
        return mailAddress;
    }

    public void setMailAddress(String mailAddress) {
        this.mailAddress = mailAddress;
    }
}
```

#### 7.5.4.5.USUARIOS

```
package com.cybertools.collegeModel;

public class ModelUsers {
```

```

private String mail, pass;

public String getMail() {
    return mail;
}

public void setMail(String mail) {
    this.mail = mail;
}

public String getPass() {
    return pass;
}

public void setPass(String pass) {
    this.pass = pass;
}
}

```

## 7.5.5. SERVLETS

### 7.5.5.1. SERVLET CARRERAS

```

package com.cybertools.collegeServlet;

//author jpjar

import com.cybertools.collegeController.*;
import com.cybertools.collegeModel.ModelCareers;
import com.google.gson.*;
import java.io.*;
import java.util.List;
import jakarta.servlet.annotation.*;
import jakarta.servlet.http.*;

@WebServlet(name = "CareersServlet", urlPatterns = "/careers")
@MultipartConfig

public class ServletCareers extends HttpServlet {

    private static final Gson objGson = new GsonBuilder().serializeNulls().create();
    private static final DAO<ModelCareers> dao = new ControllerCareers();

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {
        response.setContentType("application/json");
        PrintWriter out = response.getWriter();
        List<ModelCareers> list = dao.read();
        String data = objGson.toJson(list);
        out.write(data);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException {
        String formData = objGson.toJson(request.getParameterMap());
        formData = formData.replaceAll("\\\\\"", "");
        ModelCareers mc = objGson.fromJson(formData, ModelCareers.class);
        boolean create = dao.create(mc);
        if(create){
            response.setStatus(HttpServletResponse.SC_OK);
        }else{
            response.sendError(HttpServletResponse.SC_CONFLICT);
        }
    }

    protected void doPut(HttpServletRequest request, HttpServletResponse response) throws IOException{
        String formData = objGson.toJson(request.getParameterMap());
        formData = formData.replaceAll("\\\\\"", "");
        ModelCareers mc = objGson.fromJson(formData, ModelCareers.class);
        boolean update = dao.update(mc);
        if(update){
            response.setStatus(HttpServletResponse.SC_OK);
        }else{
            response.sendError(HttpServletResponse.SC_CONFLICT);
        }
    }
}

```

```

protected void doDelete(HttpServletRequest request, HttpServletResponse response) throws IOException{
    String formData = objGson.toJson(request.getParameterMap());
    formData = formData.replaceAll("[\\[\\]]", "");
    ModelCareers mc = objGson.fromJson(formData, ModelCareers.class);
    boolean del = dao.delete(mc);
    if(del){
        response.setStatus(HttpServletRequest.SC_OK);
    }else{
        response.sendError(HttpServletRequest.SC_CONFLICT);
    }
}
}

```

### 7.5.5.2.SERVLET LOGIN

```

● ● ●

package com.cybertools.collegeServlet;

import com.cybertools.collegeController.ControllerLogin;
import com.cybertools.collegeController.DAOLogin;
import com.cybertools.collegeModel.ModelUsers;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import jakarta.servlet.annotation.*;
import jakarta.servlet.http.*;
import java.io.IOException;

@WebServlet(name = "LoginServlet", urlPatterns = "/login")
@MultipartConfig
public class ServletLogin extends HttpServlet {

    private static final Gson objGson = new GsonBuilder().serializeNulls().create();
    private static final DAOLogin<ModelUsers> dao = new ControllerLogin();

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {
        String formData = objGson.toJson(request.getParameterMap());
        formData = formData.replaceAll("[\\[\\]]", "");
        ModelUsers mu = objGson.fromJson(formData, ModelUsers.class);
        boolean login = dao.login(mu);
        if(login){
            HttpSession session = request.getSession();
            session.setMaxInactiveInterval(10*60);
            response.sendRedirect("Introduction.jsp");
        }else {
            response.sendError(HttpServletRequest.SC_CONFLICT);
        }
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException {
        String formData = objGson.toJson(request.getParameterMap());
        formData = formData.replaceAll("[\\[\\]]", "");
        ModelUsers mu = objGson.fromJson(formData, ModelUsers.class);
        boolean register = dao.register(mu);
        if(register){
            HttpSession session = request.getSession();
            session.setMaxInactiveInterval(10*60);
            response.sendRedirect("Introduction.jsp");
        }else {
            response.sendError(HttpServletRequest.SC_CONFLICT);
        }
    }
}

```

### 7.5.5.3.SERVLET PROCEDIMIENTOS

```
package com.cybertools.collegeServlet;

import com.cybertools.collegeController.ControllerProcedures;
import com.cybertools.collegeController.DAOProcedures;
import com.cybertools.collegeModel.ModelStudents;
import com.google.gson.*;
import jakarta.servlet.annotation.MultipartConfig;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.*;
import java.io.IOException;
import java.io.PrintWriter;

@WebServlet(name = "ProceduresServlet", urlPatterns = "/procedures/correoEst")
@MultipartConfig
public class ServletProcedures extends HttpServlet{

    private static final Gson objGson = new GsonBuilder().serializeNulls().create();

    private static final DAOProcedures<ModelStudents> dao = new ControllerProcedures();

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException {
        String formData = objGson.toJson(request.getParameterMap());
        formData = formData.replaceAll("[\\[\\]]", "");
        ModelStudents ms = objGson.fromJson(formData, ModelStudents.class);
        String result = dao.correoEst(ms);
        PrintWriter out = response.getWriter();
        if(result != null){
            response.setStatus(HttpServletResponse.SC_OK);
            out.write(result);
        }else{
            response.sendError(HttpServletResponse.SC_CONFLICT);
        }
    }
}
```

### 7.5.5.4.SERVLET ESTUDIANTES

```
package com.cybertools.collegeServlet;
//author jpiar

import com.cybertools.collegeController.*;
import com.cybertools.collegeModel.ModelStudents;
import com.google.gson.*;
import java.io.*;
import java.util.List;
import jakarta.servlet.annotation.*;
import jakarta.servlet.http.*;

@WebServlet(name = "StudentsServlet", urlPatterns = "/students")
@MultipartConfig

public class ServletStudents extends HttpServlet{

    private static final Gson objGson = new GsonBuilder().serializeNulls().create();
    private static final DAO<ModelStudents> dao = new ControllerStudents();

    //Obtiene datos del servidor
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {
        response.setContentType("application/json");
        PrintWriter out = response.getWriter();
        List<ModelStudents> list = dao.read();
        String data = objGson.toJson(list);
        out.write(data);
    }
}
```

```

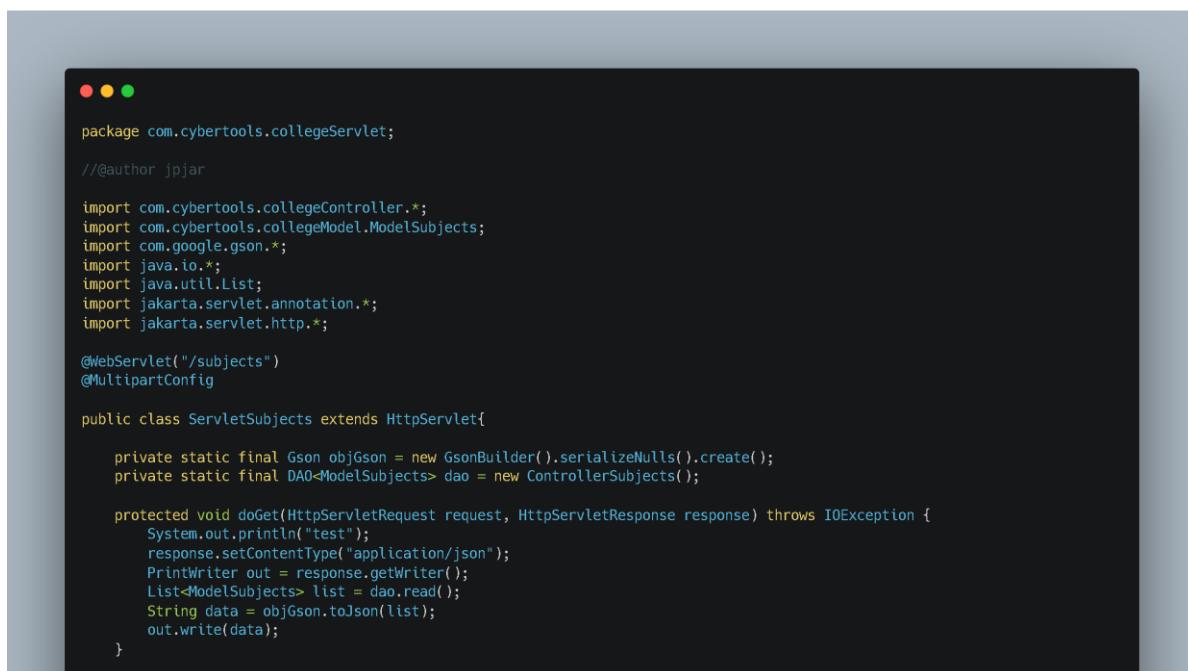
//Crea datos en el server
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException {
    String formData = objGson.toJson(request.getParameterMap()); //optiene todos los datos de una sola vez
    formData = formData.replaceAll("[\\n\\r]", ""); //reemplaza simbolos de request
    ModelStudents mSt = objGson.fromJson(formData, ModelStudents.class); //se convierte al objeto para eso se usa el
.class
    boolean create = dao.create(mSt);
    if(create){
        response.setStatus(HttpServletResponse.SC_OK);
    }else{
        response.sendError(HttpServletResponse.SC_CONFLICT);
    }
}

//Actualiza datos del server
protected void doPut(HttpServletRequest request, HttpServletResponse response) throws IOException{
    String formData = objGson.toJson(request.getParameterMap());
    formData = formData.replaceAll("[\\n\\r]", "");
    ModelStudents mSt = objGson.fromJson(formData, ModelStudents.class);
    boolean update = dao.update(mSt);
    if(update){
        response.setStatus(HttpServletResponse.SC_OK);
    }else{
        response.sendError(HttpServletResponse.SC_CONFLICT);
    }
}

protected void doDelete(HttpServletRequest request, HttpServletResponse response) throws IOException{
    String formData = objGson.toJson(request.getParameterMap());
    formData = formData.replaceAll("[\\n\\r]", "");
    ModelStudents mSt = objGson.fromJson(formData, ModelStudents.class);
    boolean del = dao.delete(mSt);
    if(del){
        response.setStatus(HttpServletResponse.SC_OK);
    }else{
        response.sendError(HttpServletResponse.SC_CONFLICT);
    }
}
}

```

### 7.5.5.5.SERVLET MATERIAS



```

package com.cybertools.collegeServlet;

//@author jjar

import com.cybertools.collegeController.*;
import com.cybertools.collegeModel.ModelSubjects;
import com.google.gson.*;
import java.io.*;
import java.util.List;
import jakarta.servlet.annotation.*;
import jakarta.servlet.http.*;

@WebServlet("/subjects")
@MultiPartConfig

public class ServletSubjects extends HttpServlet{

    private static final Gson objGson = new GsonBuilder().serializeNulls().create();
    private static final DAO<ModelSubjects> dao = new ControllerSubjects();

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {
        System.out.println("test");
        response.setContentType("application/json");
        PrintWriter out = response.getWriter();
        List<ModelSubjects> list = dao.read();
        String data = objGson.toJson(list);
        out.write(data);
    }
}

```

```

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException {
    String formData = objGson.toJson(request.getParameterMap()); //optiene todos los datos de una sola vez
    formData = formData.replaceAll("[\\n\\r]", ""); //reemplaza simbolos de request
    ModelSubjects mSt = objGson.fromJson(formData, ModelSubjects.class); //se convierte al objeto para eso se usa el
    .class
    boolean create = dao.create(mSt);
    if(create){
        response.setStatus(HttpServletRequest.SC_OK);
    }else{
        response.sendError(HttpServletRequest.SC_CONFLICT);
    }
}

protected void doPut(HttpServletRequest request, HttpServletResponse response) throws IOException{
    String formData = objGson.toJson(request.getParameterMap());
    formData = formData.replaceAll("[\\n\\r]", "");
    ModelSubjects mSt = objGson.fromJson(formData, ModelSubjects.class);
    boolean update = dao.update(mSt);
    if(update){
        response.setStatus(HttpServletRequest.SC_OK);
    }else{
        response.sendError(HttpServletRequest.SC_CONFLICT);
    }
}

protected void doDelete(HttpServletRequest request, HttpServletResponse response) throws IOException{
    String formData = objGson.toJson(request.getParameterMap());
    formData = formData.replaceAll("[\\n\\r]", "");
    ModelSubjects mSt = objGson.fromJson(formData, ModelSubjects.class);
    boolean del = dao.delete(mSt);
    if(del){
        response.setStatus(HttpServletRequest.SC_OK);
    }else{
        response.sendError(HttpServletRequest.SC_CONFLICT);
    }
}
}

```

### 7.5.5.6.SERVLET PROFESORES

```

● ● ●

package com.cybertools.collegeServlet;
//@author jpjar

import com.cybertools.collegeController.*;
import com.cybertools.collegeModel.ModelTeachers;
import com.google.gson.*;
import java.io.*;
import java.util.List;
import jakarta.servlet.annotation.*;
import jakarta.servlet.http.*;

@WebServlet(name = "TeachersServlet", urlPatterns = "/teachers")
@MultiPartConfig

public class ServletTeachers extends HttpServlet{

    private static final Gson objGson = new GsonBuilder().serializeNulls().create();
    private static final DAO<ModelTeachers> dao = new ControllerTeachers();

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {
        response.setContentType("application/json");
        PrintWriter out = response.getWriter();
        List<ModelTeachers> list = dao.read();
        String data = objGson.toJson(list);
        out.write(data);
    }
}

```

```

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException {
    String formData = objGson.toJson(request.getParameterMap());
    formData = formData.replaceAll("[\\n\\r]", "");
    ModelTeachers mt = objGson.fromJson(formData, ModelTeachers.class);
    boolean create = dao.create(mt);
    if(create){
        response.setStatus(HttpServletRequest.SC_OK);
    }else{
        response.sendError(HttpServletRequest.SC_CONFLICT);
    }
}

protected void doPut(HttpServletRequest request, HttpServletResponse response) throws IOException{
    String formData = objGson.toJson(request.getParameterMap());
    formData = formData.replaceAll("[\\n\\r]", "");
    ModelTeachers mt = objGson.fromJson(formData, ModelTeachers.class);
    boolean update = dao.update(mt);
    if(update){
        response.setStatus(HttpServletRequest.SC_OK);
    }else{
        response.sendError(HttpServletRequest.SC_CONFLICT);
    }
}

protected void doDelete(HttpServletRequest request, HttpServletResponse response) throws IOException{
    String formData = objGson.toJson(request.getParameterMap());
    formData = formData.replaceAll("[\\n\\r]", "");
    ModelTeachers mt = objGson.fromJson(formData, ModelTeachers.class);
    boolean del = dao.delete(mt);
    if(del){
        response.setStatus(HttpServletRequest.SC_OK);
    }else{
        response.sendError(HttpServletRequest.SC_CONFLICT);
    }
}
}

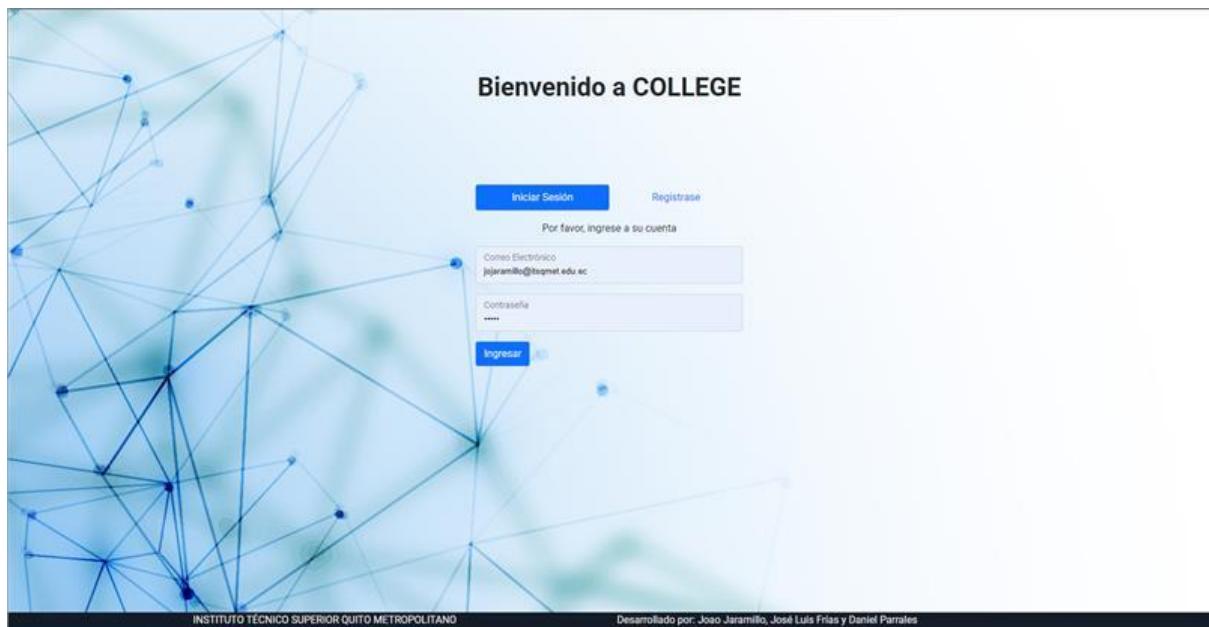
```

### 7.5.6. DEPENDENCIAS USADAS



## 8. ANÁLISIS DE RESULTADOS

Se realizó el sitio web como se puede apreciar en las siguientes imágenes.



Se creó un diseño simple para el ingreso de datos en la sesión, con sus respectivos campos de contraseña y correo electrónico. Si no se tiene una cuenta es necesario registrarse.

A screenshot of the application's main content area. At the top is a navigation bar with the ITSM logo and links for "Estudiantes", "Profesores", "Carreras &amp; Asignaturas", "Procedimientos", and "Logout". The main content has a blue network background. It includes sections for "Introducción", "Objetivo General", and "Objetivos Específicos". The "Introducción" section contains a paragraph about the application. The "Objetivo General" section lists the goal of developing a web application using Netbeans and MySQL. The "Objetivos Específicos" section lists four specific goals related to program characteristics, execution, and data storage. The footer is identical to the login page.

Implementamos un sistema de carrusel para tener mejor organización entre las diferentes páginas las cuales albergaran información. También se creó footer para todas las páginas. En el inicio tenemos la introducción y los objetivos de nuestro proyecto.

**Control para Estudiantes**

Insertar nuevo Estudiante

NUI	Nombre	Apellido	Dirección	Correo	Teléfono	Acción
0101642486	MARIA KERLI	QUITO RODAS	LA CORUÑA	kerlyqr@hotmail.com	0984056831	Actualizar Eliminar
0401402215	ARTURO EDUARDO	ROMO ORBE	MIRAFLORES	arturoromo2009@gmail.com	0999806431	Actualizar Eliminar
0503936444	CRISTINA ESPERANZA	CAMPUZANO ÁRRIAGA	CUMBAYA	cristinacampuzano3@gmail.com	0986342081	Actualizar Eliminar
0801826611	MARIA JOSE	MIENDEZ NEWBALL	CUMBAYA	mariajosemendez85@hotmail.com	0959799263	Actualizar Eliminar
0906067822	GUSTAVO HECTOR	MACIAS ROMERO	LA CORUÑA	gusmac@hotmail.es	0993518007	Actualizar Eliminar
0910654498	RODRIGO HERNAN	CARRILLO QUINTANA	CUMBAYA	rodcar17@hotmail.com	0992015056	Actualizar Eliminar
0913407110	LAURA BEATRIZ	OCHOA GUERRERO	TUMBACO	fernandoren29@hotmail.com	0985423545	Actualizar Eliminar

INSTITUTO TÉCNICO SUPERIOR QUITO METROPOLITANO Desarrollado por: Joao Jaramillo, José Luis Frías y Daniel Parnales.

**Control para Docentes**

Insertar nuevo Docente

NUI	Nombre	Apellido	Correo	Acción
0201512290	SIXTO RENAN	MONAR VERDEZOTO	rmonar2009@hotmail.com	Actualizar Eliminar
0501695366	FRANCISCO ROLANDO	SAN LUCAS	pastorsanlucas@gmail.com	Actualizar Eliminar
0502449002	IVAN FERNANDO	VENEGAS ARMENDARIZ	femanarme@hotmail.com	Actualizar Eliminar
0600778591	NELLY HORTENCIA	GALARAGA BRITO	nelly.galaraga@gmail.com	Actualizar Eliminar
0601849094	VICTOR MANUEL	ECHEVERRIA INZUASTI	victormanuelecheverria1964@gmail.com	Actualizar Eliminar
0602146532	CARLOS ADRIANO	SALAZAR CARDOSO	cesenval@hotmail.com	Actualizar Eliminar
0604172371	GIOVANNY	MONTUFAR ECHEVERRIA	gmontufare8@gmail.com	Actualizar Eliminar

INSTITUTO TÉCNICO SUPERIOR QUITO METROPOLITANO Desarrollado por: Joao Jaramillo, José Luis Frías y Daniel Parnales.

Se crearon tablas para la organización de los estudiantes del instituto, donde podemos añadir, modificar, eliminar información de los estudiantes de los diferentes campos. Se implementó un sistema de búsqueda, para que nuestra página se eficaz al momento de visualizar los datos.

ID	Carrera	Acción
1	DESARROLLO DE SOFTWARE	Actualizar Eliminar
2	REDES Y TELECOMUNICACIONES	Actualizar Eliminar
3	ADMINISTRACIÓN DE EMPRESAS	Actualizar Eliminar
4	EDUCACIÓN INICIAL	Actualizar Eliminar
5	TALENTO HUMANO	Actualizar Eliminar
6	ADAPTACIONES WEB I	Actualizar Eliminar
7	MARKETING DIGITAL I	Actualizar Eliminar

ID	Asignatura	Acción
2	PROGRAMACIÓN ORIENTADA A OBJETOS I	Actualizar Eliminar
3	ÉTICA	Actualizar Eliminar
4	BASE DE DATOS II	Actualizar Eliminar
5	SISTEMAS OPERATIVOS	Actualizar Eliminar
6	ADAPTACIONES WEB	Actualizar Eliminar
7	BASE DE DATOS II	Actualizar Eliminar
8	PROGRAMACIÓN ORIENTADA A OBJETOS II	Actualizar Eliminar

Se gestionaron las carreras y asignatura en una página, para una mejor presentación. Se implementó un sistema de búsqueda y donde también podremos añadir, modificar y eliminar los respectivos campos.

## 9. VERIFICACION DE CALIDAD DE SOFTWARE

Mediante una web de validación de html ingresamos nuestras líneas de código y nos mostraron algunos errores, los cuales no teníamos presente.

1. **Error** carácter incorrecto ` después de <. Causa probable: Sin escape <. Intenta escapar como &lt;`.  
En la línea 1 , columna 2  
[<%page content%]

2. **Error** se encontraron caracteres que no son espacios sin ver primero un tipo de documento. Esperado <!DOCTYPE html>  
De la línea 1 , columna 1 , a la línea 1 , columna 2  
[<%page %]

3. **Error** al elemento `head` le falta una instancia necesaria del elemento secundario `title`.  
De la línea 1 , columna 1 , a la línea 1 , columna 2  
[<%page %]  
Modelo de contenido para el elemento `head`:  
Si el documento es `un iframe` o si la información del título está disponible en un protocolo de nivel superior. Cero o más elementos de `contenido de metadatos`, de los cuales no más de uno es un `title` elemento y no más de uno es un `base` elemento.  
De lo contrario: uno o más elementos del `contenido de metadatos`, de los cuales exactamente uno es un `title` elemento y no más de uno es un `base` elemento.

4. **Error** tipo de documento extraviado.  
De la línea 2 , columna 1 , a la línea 2 , columna 15  
[UTF-8" ><!DOCTYPE html></html>]

5. **Error** etiqueta de inicio extraviada `html`.  
De la línea 3 , columna 1 , a la línea 3 , columna 6  
[YPE html><html><head>

6. **Error** etiqueta de inicio extraída `head`.  
De la línea 4 , columna 1 , a la línea 4 , columna 6  
`<!--<html>--><head>-->`

7. **Error** atributo `http-equiv` no permitido en el elemento `meta` en este punto.  
De la línea 5 , columna 5 , a la línea 5 , columna 7  
`<head>--> <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">-->`  
 Atributos para el elemento `meta`:  
 Attributos globales  
`name`— Nombre de metadatos  
`http-equiv`— Directiva Pragma  
`content`— Valor del elemento  
`charset`— Declaración de codificación de caracteres  
`media`— Medios aplicables

8. **Error** al elemento `meta` le falta uno o más de los siguientes atributos: `itemprop`, `property`.  
De la línea 5 , columna 5 , a la línea 5 , columna 7  
`<head>--> <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">-->`  
 Atributos para el elemento `meta`:  
 Attributos globales  
`name`— Nombre de metadatos  
`http-equiv`— Directiva Pragma  
`content`— Valor del elemento

9. **Error** Elemento `title` no permitido como hijo de elemento `body` en este contexto. (Suprimiendo más errores de este subárbol).  
De la línea 6 , columna 5 , a la línea 6 , columna 11  
`<!--> <title>Introd`  
 Contextos en los que `title` se puede utilizar el elemento:  
 En un `head` elemento que no contiene otros `title` elementos.  
 Modelo de contenido para el elemento `body`:  
`Contenido de flujo`.

10. **Error** carácter incorrecto `%` después de `<`. Causa probable: Sin escape `<`. Intenta escapar como `&lt;`.  
En la línea 7 , columna 6  
`</title>--> <%include file=`

11. **Información** La barra inclinada final en los elementos vacíos [no tiene ningún efecto](#) e [interactúa mal con los valores de atributo sin comillas](#).  
De la línea 8 , columna 5 , a la línea 8 , columna 10  
`1" %>--> <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css"/>-->`

12. **Información** La barra inclinada final en los elementos vacíos [no tiene ningún efecto](#) e [interactúa mal con los valores de atributo sin comillas](#).  
De la línea 9 , columna 5 , a la línea 9 , columna 6  
`ss"/>--> <link href="Styles/Styles.css" rel="stylesheet" type="text/css"/>--></hea`

13. **Error** etiqueta final extraída `head`.  
De la línea 10 , columna 1 , a la línea 10 , columna 7  
`xt/css"/>--></head>--><body>`

14. **Error** la etiqueta de inicio `body` , pero ya estaba abierto un elemento del mismo tipo.  
De la línea 11 , columna 1 , a la línea 11 , columna 22  
`>--></head>--><body id="background">--><div>`

15. **Error fatal** no se puede recuperar después del último error. Cualquier otro error será ignorado.  
De la línea 11 , columna 1 , a la línea 11 , columna 22  
`>--></head>--><body id="background">--><div>`

## 10. CONCLUSIONES

Después de lo expuesto podemos concluir que independientemente que nuestro producto se este ejecutando de manera correcta, puede existir algunos errores en el código que puede ayudarnos a agilitar los procesos, funcionalidades de la misma.

## 11. RECOMENDACIONES

- Para que nuestro producto este bien conformado hay que implementar una arquitectura de diseño para distribuir bien las funcionalidades de este
- Distribuir la información HTML en diferentes páginas, para mantener una buena presentación del sitio web y que sea amigable para el usuario.
- Una vez tengamos nuestro producto validar la calidad de programa o aplicación estamos entregando para que a futuro si necesitamos algún cambio no tengamos complicaciones.

## 12. BIBLIOGRAFÍA

Rodrigo González González y Jorge Jimeno Bernal. (2012). *Check list / Listas de chequeo: ¿Qué es un checklist y cómo usarlo?* Obtenido de Pdcahome:

<https://www.pdcahome.com/check-list/>

Chacon, S., & Straub, B. (2021). *Pro Git*. Apress.

desarrollowe6. (28 de Julio de 2020). *Manuales*. Obtenido de  
<https://desarrolloweb.com/articulos/que-es-mvc.html>

Gabriel, B. U. (2016). *Introducción a la informática*. Patria.

Genos. (s.f.). *Cloud Services*. Obtenido de <https://genos.es/tomcat-soporte/>  
GitHub. (29 de julio de 2021). *docs.github*. Obtenido de [docs.github.com: https://docs.github.com/en/get-started/onboarding/getting-started-with-your-github-account](https://docs.github.com/en/get-started/onboarding/getting-started-with-your-github-account)

IMMUNE Tecnology Institute. (13 de abril de 2022). Obtenido de  
<https://immune.institute/blog/que-es-netbeans/>

Keep Coding. (11 de abril de 2022). *Home Blog*. Obtenido de <https://keepcoding.io/blog/que-es-mysql-workbench/>

Lopez Pellicer, F., Latre, M., Nogueras, J., & Zarazaga, J. (2015). *GitHub como herramienta docente*. Andorra: Universidad de Zaragoza.

MDN Contributors. (05 de septiembre de 2022). *Web Docs*. Obtenido de [https://developer.mozilla.org/es/docs/Learn/Getting\\_started\\_with\\_the\\_web/JavaScript\\_basics](https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/JavaScript_basics)

Santos, C. (2015). Seguridad informática. En *Software de Seguridad* (pág. 44). Madrid, España: RA-MA.

theastrologypage. (2023). *Blog*. Obtenido de <https://es.theastrologypage.com/html-validator>

## ANEXO

Link del repositorio del Proyecto en GitHub: <https://github.com/LuiggiMH/College.git>