

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



INSTITUTO TECNOLÓGICO SUPERIOR
QUITO METROPOLITANO
FORMANDO PROFESIONALES DE ÉLITE
ITSQMET 2021

EVIDENCIAR EL PROCESO DE
APRENDIZAJE POR PARTE DE
LOS ESTUDIANTES

WWW.ITSQMET.EDU.EC



INSTITUTO TECNOLÓGICO SUPERIOR QUITO METROPOLITANO



INSTITUTO TECNOLÓGICO SUPERIOR
QUITO METROPOLITANO

FORMANDO PROFESIONALES DE ÉLITE

ITSQM^{ET}

**PROYECTO UNIFICADO DE LAS ASIGNATURAS:
PROGRAMACION ORIENTADA A OBJETOS II
ADAPTACIONES WEB I
BASE DE DATOS II**

DOCENTES:

ING. CRISTIAN NARANJO

ING. VERONICA ZAPATA

ESTUDIANTES:

JIMMY DANIEL PARRALES IDROVO

JOAO PAUL JARAMILLO VILLALBA

JOSÉ LUIS FRÍAS HERNÁNDEZ

ABRIL 2022 – SEPTIEMBRE 2022

CONTENIDO

PROYECTO FINAL – “COLLEGE”	6
1. INTRODUCCIÓN AL PROYECTO DE APLICACIÓN.....	6
2. OBJETIVO GENERAL DEL PROYECTO.....	6
3. OBJETIVOS ESPECÍFICOS DEL PROYECTO	6
4. PLANTEAMIENTO DEL PROBLEMA.....	7
5. JUSTIFICACION	7
6. MARCO TEORICO	7
6.1. JAVA.....	7
6.1.1. JAVASCRIPT	8
6.2. HTML.....	8
6.3. CSS.....	9
6.4. JAKARTA EE.....	10
6.5. GITHUB.....	10
6.6. BASE DE DATOS	11
6.6.1.1. ¿QUÉ ES UNA BASE DE DATOS?.....	11
6.6.1.2. ¿QUÉ ES EL LENGUAJE SQL?	12
6.6.1.3. ¿QUÉ ES MYSQL?	12
7. HERRAMIENTAS	12
7.1. NETBEANS	12
7.2. MAVEN	13
7.3. WORKBECH	14
7.4. TOMCAT	14
7.5. BOOTSTRAP 5.....	15
7.6. API FETCH.....	15

7.7.	API JSON	16
8.	DESARROLLO Y/O IMPLEMENTACION	17
8.1.	LEVANTAMIENTO DE LA BASE DE DATOS	17
8.2.	DIAGRAMA ENTIDAD-RELACION.....	19
8.3.	DIAGRAMA DE ARQUITECTURA	19
8.4.	IMPLEMENTACION DE LA BASE DE DATOS	20
8.5.	DIAGRAMA DE ACTIVIDADES.....	22
8.6.	CODIFICACION DE LA PÁGINA	23
8.6.1.	CÓDIGO HTML EN NETBEANS CON JAVA	23
8.6.1.1.	INTRODUCCIÓN DEL SITIO WEB	23
8.6.1.2.	LOGIN DEL SITIO WEB	24
8.6.1.3.	VISTA DE CARRERAS Y MATERIAS DE LA PÁGINA	27
8.6.1.4.	VISTA DE ESTUDIANTES DEL SITIO WEB	29
8.6.1.5.	VISTA DE PROFESORES DEL SITIO WEB.....	31
8.6.2.	CONEXIÓN CON LA BASE DE DATOS.....	32
8.6.3.	CONTROLADORES	33
8.6.3.1.	CONTROLADOR DE CARRERAS	33
8.6.3.2.	CONTROLADOR DE LOGIN.....	34
8.6.3.3.	CONTROLADOR PROCEDURES	35
8.6.3.4.	CONTROLADOR ESTUDIANTES	35
8.6.3.5.	CONTROLADOR DE MATERIAS.....	37
8.6.3.6.	CONTROLADOR DE PROFESORES	38
8.6.4.	MODELOS	39
8.6.4.1.	CARRERA.....	39
8.6.4.2.	ESTUDIANTES	39

8.6.4.3.	MATERIAS	40
8.6.4.4.	PROFESORES.....	41
8.6.4.5.	USUARIOS.....	41
8.6.5.	SERVLETS	42
8.6.5.1.	SERVLET CARRERAS.....	42
8.6.5.2.	SERVLET LOGIN.....	43
8.6.5.3.	SERVLET PROCEDIMIENTOS	44
8.6.5.4.	SERVLET ESTUDIANTES	44
8.6.5.5.	SERVLET MATERIAS.....	45
8.6.5.6.	SERVLET PROFESORES	46
8.6.6.	DEPENDENCIAS USADAS	47
9.	ANÁLISIS DE RESULTADOS.....	47
10.	CONCLUSIONES	50
11.	RECOMENDACIONES.....	50
12.	BIBLIOGRAFÍA	51
13.	ANEXOS	53

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

Desarrollar un Página Web “College” que nos permita la administración de datos para una Institución Educativa mediante la utilización de las diferentes herramientas y los conocimientos adquiridos en las diferentes asignaturas a lo largo de la carrera.

3. OBJETIVOS ESPECÍFICOS DEL PROYECTO

- Desarrollar una interfaz gráfica en HTML intuitiva y amigable para los usuarios.
- Adicionar a la página tecnología JSF y funcionalidades a través de JavaScript, que permitan una mejor interacción a los usuarios.
- Implementar una base datos para un mejor control de registros, añadiendo diferentes roles y administración de credenciales, también funciones como triggers y procedimientos almacenados.

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.

Sin embargo, muchas veces por falta de recursos no pueden o no ven necesario la implementación de una página web y una base de datos que ayude a la institución a manejar correcta y ordenadamente los datos internos. Incluso si no hay una vista amistosa y de fácil uso para los usuarios puede generar una desazón entre los mismos y provocar la pérdida de estudiantes ya matriculados o interesados en pertenecer a la institución.

5. JUSTIFICACION

Si queremos que una institución educativa crezca, aparte del buen nivel de enseñanza, debe tener una página web acorde a la calidad de servicio que brinda.

Dicha página además de ser amigable a los usuarios, debe ser bien estructurada en todas sus vistas y conectada correctamente a una base de datos que administre la información, genere reportes, procedimientos y funciones en todas sus tablas.

6. MARCO TEORICO

6.1.JAVA

Java es un lenguaje de programación ampliamente utilizado para codificar aplicaciones web. Es Lenguaje de Alto Nivel, java sirve para crear aplicaciones y procesos en una gran diversidad de dispositivos. se basa en programación orientada a objetivos, permite ejecutar

un mismo programa en diversos sistemas operativos y ejecutar el código en sistemas remotos de manera segura.

Debido a que Java es un lenguaje versátil y de uso gratuito, crea software localizado y distribuido. Algunas características son: Es simple, Orientado a objetos, Independiente a la Plataforma, Recolector de basura, Seguro y sólido, Multihilo. (Oracle, 2014)

6.1.1. JAVASCRIPT

JavaScript es un robusto lenguaje de programación que se puede aplicar a un documento HTML y usarse para crear interactividad dinámica en los sitios web. Puedes hacer casi cualquier cosa con JavaScript. Se puede empezar con pequeñas cosas como carruseles, galerías de imágenes, diseños fluctuantes, y respuestas a las pulsaciones de botones. Con más experiencia, serás capaz de crear juegos, animaciones 2D y gráficos 3D, aplicaciones integradas basadas en bases de datos y mucho más

JavaScript por sí solo es bastante compacto, aunque muy flexible, y los desarrolladores han escrito gran cantidad de herramientas encima del núcleo del lenguaje JavaScript, desbloqueando una gran cantidad de funcionalidad adicional con un mínimo esfuerzo. (MDN Contributors, 2022)

6.2. HTML

HTML (Lenguaje de marcado de hipertexto o HyperText Markup Language por sus siglas en inglés) es un lenguaje descriptivo que especifica la estructura de las páginas web. HTML nos permite definir los nuevos estándares de desarrollo web, modificando el código

existente para solucionar problemas y actualizándolo a las nuevas necesidades de hoy en día.

La versión utilizada en este proyecto es HTML5, la cual nos permite crear la estructura de una página web, por así decirlo, su esqueleto, donde vamos a incorporar posteriormente, imágenes, textos, vídeos y todo tipo de material multimedia de forma que se pueda visualizar correctamente.

A la hora de desarrollar aplicaciones web en HTML5, la gran ventaja es la adaptabilidad: la página web va a ser accesible desde ordenador, un móvil o una tablet, y HTML5 es compatible con los diseños adaptativos para que la web reconozca el dispositivo desde el que se está accediendo y se adapte para ofrecer la mejor funcionalidad. (Grupo Conforsa, 2021)

6.3.CSS

CSS, de las siglas en inglés Cascading Style Sheets (Hojas de Estilo en Cascada), es un lenguaje declarativo que controla el aspecto de las páginas web en el navegador. El navegador aplica declaraciones de estilo CSS a los elementos seleccionados para exhibirlos correctamente.

Una declaración de estilos contiene las propiedades y sus respectivos valores, los cuales determinan cómo se verá una página web.

CSS es una de las tres principales tecnologías web, junto con HTML y JavaScript. CSS usualmente le da estilo a los (elementos HTML), pero también puede ser utilizado con otros lenguajes de marcado como SVG o XML.

El término "en cascada" se refiere a las reglas que determinan cómo los selectores son jerarquizados al cambiar la apariencia de una página web. Esta es una característica muy importante, ya que un sitio web complejo puede contener miles de reglas CSS. (MDN Contributors, 2022)

6.4.JAKARTA EE

Es una extensión moderna de java que ofrece aplicaciones actuales para satisfacer al desarrollador, hasta ahora ha liberado la versión 8 de esta. Es desarrollada por Oracle siendo una herramienta Open Source, código abierto.

Jakarta tiene especificaciones como: Jakarta Enterprise Java Beans, Jakarta Persistence y Jakarta Restful Web Services. (IBM, 2022)

6.5.GITHUB

GitHub es uno 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.

6.6.BASE DE DATOS

6.6.1.1.¿QUÉ ES UNA BASE DE DATOS?

Una base de datos se usa para el almacenamiento de la información de forma organizada según el libro de Bases de datos de Mercedes Marqués “La base de datos es un conjunto de datos almacenados en memoria externa que están organizados mediante una estructura de datos.” (**Marqués, 2009**) Lo cual ha mejorado la automatización de los datos ingresados, cambiar la información, como también borrarla. Además de esto se mejora la seguridad de la información creando BackUps (Respaldos).

La información que se guarda en una base de datos puede ser compartida de manera segura, siendo el usuario principal el administrador quién tendrá todos los privilegios de esta, a su vez, el administrador puede otorgar ciertos privilegios necesarios a otros usuarios para la administración de la información, según como el propietario o administrador deseé. (**Marqués, 2009, pág. 1, pp. 5**)

6.6.1.2.¿QUÉ ES EL LENGUAJE SQL?

SQL que en sus siglas significa Lenguaje de consulta Estructurado, uno de los lenguajes que más se usa en la gestión de datos, siendo por el uso de sistemas actuales y su alta operatividad en funciones. Operaciones como: Select Insert, Update, Delete, entre otras. (**Marqués, 2009**)

Este lenguaje fue desarrollado por la empresa IBM, en el año 1970. Siendo útil para interactuar con la información, implementado mayormente en negocios y empresas. (**Quintana, Marqués, & J.I. Aliga, 2014**)

6.6.1.3.¿QUÉ ES MYSQL?

Es un sistema de gestión de datos relacionales, desarrollado por Oracle, es de código abierto, lo que quiere decir que su uso es libre del usuario que lo adquiere. Admite lenguajes de desarrollo como Pitón, PHP, Java, C++, entre otros.

Las características que definen a este software son, es fácil de usar, rápido y confiable. Es por ello que este ha impulsado a plataformas como Facebook, Twitter, Netflix, Uber entre otros. (**oracle, 2022**)

7. HERRAMIENTAS

7.1.NETBEANS

NetBeans es un IDE o entorno de desarrollo integrado, basado en el lenguaje Java y ejecutado en Swing.

De esta forma, NetBeans o Apache NetBeans es una aplicación de código abierto, que ha cobrado bastante popularidad en los últimos años.

Este IDE, orientado principalmente a las apps de Java, ofrece diferentes herramientas digitales como editor de texto, código, compilador, interfaz gráfica de usuario; además de un depurador.

Por otro lado, cabe destacar que NetBeans facilita la creación de aplicaciones estructuradas, ya que están basadas en un conjunto de módulos. Así, se favorece el desarrollo de las diversas funciones de una manera independiente y pudiendo también reutilizar los componentes. (IMMUNE Tecnology Institute, 2022)

7.2.MAVEN

Es una potente herramienta de gestión de proyectos que se utiliza para gestión de dependencias, como herramienta de compilación e incluso como herramienta de documentación. Es de código abierto y gratuita.

Aunque se puede utilizar con diversos lenguajes (C#, Ruby, Scala...) se usa principalmente en proyectos Java, donde es muy común ya que está escrita en este lenguaje. De hecho, frameworks Java como Spring y Spring Boot la utilizan por defecto.

Otras alternativas, como Gradle no utilizan archivos XML, sino de otro tipo, pero usan los mismos conceptos de Maven.

Con Maven se puede:

- **Gestionar las dependencias** del proyecto, para descargar e instalar módulos, paquetes y herramientas que sean necesarios para el mismo.
- **Compilar el código fuente** de la aplicación de manera automática.
- **Empaquetar el código** en archivos .jar o .zip.
- **Instalar los paquetes** en un repositorio (local, remoto o en el central de la empresa)

- **Generar documentación** a partir del código fuente.
- **Gestionar las distintas fases del ciclo de vida** de las build: validación, generación de código fuente, procesamiento, generación de recursos, compilación, ejecución de test.
(Alarcón, 2022)

7.3.WORKBECH

El software MySQL Workbench es un entorno gráfico de diseño de bases de datos, servidores, administración y mantenimiento para el sistema MySQL. Además, esta herramienta gráfica fue desarrollada y distribuida por la compañía de desarrollo de nube y locales Oracle Corporation.

MySQLWorkbench es una herramienta de acceso de diseño y modelado de bases de datos visuales para bases de datos relacionales de servidores MySQL. Facilita la creación de nuevos modelos de datos físicos y la modificación de bases de datos MySQL existentes con ingeniería inversa / avanzada y funciones de gestión de cambios.

La herramienta MySQL Workbench se encuentra disponible para su uso comercial, teniendo total compatibilidad con las versiones del servidor MySQL 5.6 en adelante. (Keep Coding, 2022)

7.4.TOMCAT

Tomcat es un contenedor open source de servlets para la implementación de Java Servlet, JavaServer Pages (JSP), Java Expression Language y Java WebSocket. Tomcat puede definirse como servidor web por sí mismo, aunque normalmente se utiliza en combinación con otros productos por ejemplo Apache, para mejorar su soporte y realizar sus

características. Tomcat puede ejecutar servlets y Java Server Pages (JSP). Al haber sido escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java. (Genos, s.f.)

7.5.BOOTSTRAP 5

Bootstrap es uno de los frameworks más populares para el desarrollo del frontend de páginas web. La herramienta proporciona plantillas para CSS y HTML que facilitan la colocación y el diseño de la página, las fuentes, los botones y los elementos de navegación, de modo que implementar con ella un diseño web moderno resulta muy sencillo.

Desarrollado originalmente para Twitter, el marco de trabajo está disponible desde hace algún tiempo como proyecto de código abierto gratuito. (IONOS, 2020)

7.6.API FETCH

La API Fetch proporciona una interfaz para recuperar recursos (incluso a través de la red). Resultará familiar a cualquiera que haya usado XMLHttpRequest, pero la nueva API ofrece un conjunto de características más potente y flexible.

Fetch ofrece una definición genérica de los objetos Request y Response (y otras cosas relacionadas con las solicitudes de red). Esto permitirá su uso donde sea necesario en un futuro, ya sea para operadores de servicios, API caché y otras cosas similares que manipulen o modifiquen las solicitudes y respuestas, o cualquier otro tipo de caso de uso que pudiera requerirle la generación de sus propias respuestas mediante programación.

También proporciona una definición para conceptos relacionados, como CORS y la semántica de encabezado HTTP origen, suplantando sus definiciones separadas en otros lugares. (MDN Contributors, 2022)

7.7. API JSON

JSON API es una convención muy utilizada para los datos estructurados. Precisamente, ha demostrado ser una oportunidad para expresar los datos de una API de manera efectiva y confiable.

JSON API es muy extenso y complicado, pero también aporta metadatos sobre los datos y permite enriquecer la información poco o no estructurada.

Esta librería estructura los JSON de la siguiente manera:

JsonElement: Esta clase representa cualquier elemento del Json que puede ser de alguno de los siguientes 4 tipos:

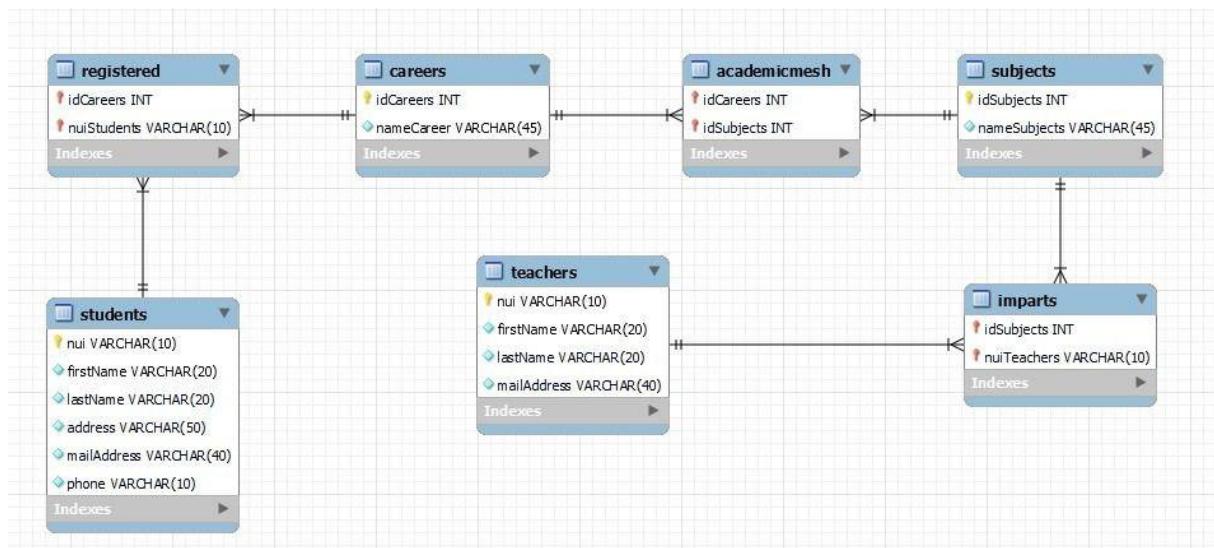
- **JsonObject:** Esta clase representa un objeto en el Json; es decir, un conjunto de pares clave-valor donde las claves son strings y los valores son cualquier otro tipo de JsonElement.
- **JsonArray:** Esta clase representa un array en el Json. Un array es una lista de JsonElements cada uno de los cuales puede ser de un tipo diferente. Se trata de una lista ordenada, por lo que el orden en que se añaden los elementos se conserva.
- **JsonPrimitive:** Esta clase representa un tipo de dato primitivo u objetos de datos simples (String, Integer, Double, etc.).
- **JsonNull:** Representa un objeto a null.

Finalmente, al igual que las demás herramientas Big Data, tiene como objetivo principal facilitar la gestión de datos y destacar el valor de la información. (Keep Coding, 2022)

8. DESARROLLO Y/O IMPLEMENTACION

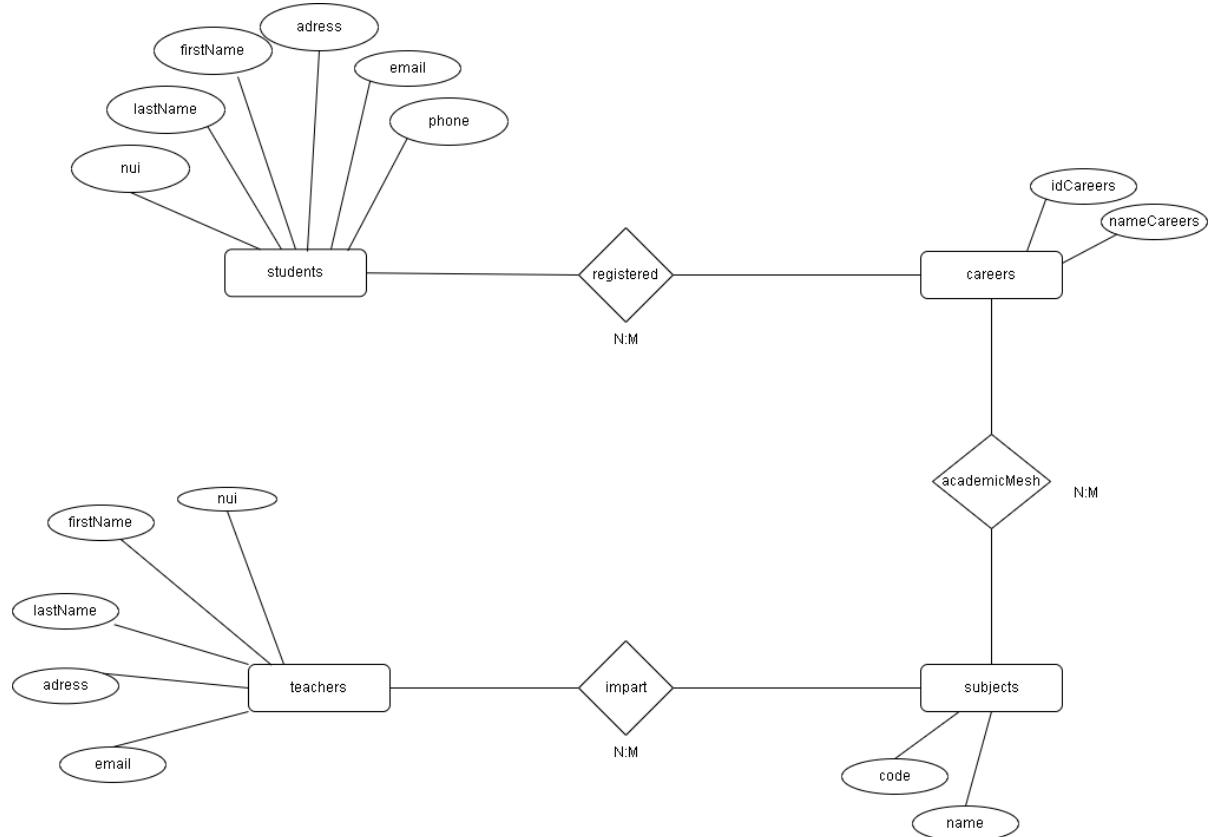
8.1.LEVANTAMIENTO DE LA BASE DE DATOS

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.

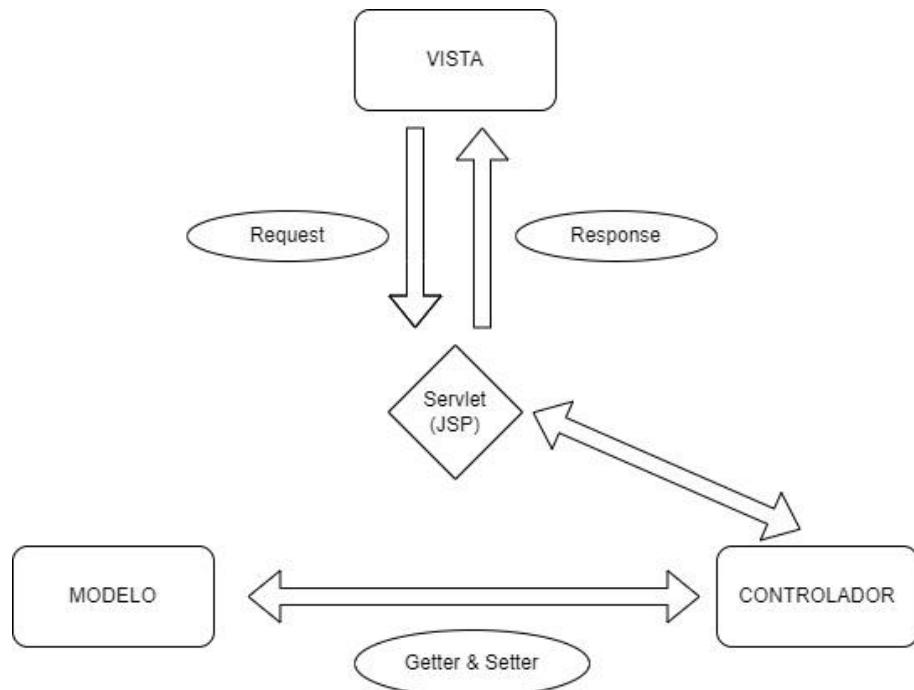


Tablas	Atributos
Students	NUI varchar(10) not null unique primary key FirstName varchar(20) int not null LastName varchar(20) int not null Address varchar (50) int not null mailAddress varchar(40) int not null Phone varchar (10) int not null
Teachers	NUI varchar(10) not null unique primary key FirstName 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
Subjetcs	idSubjets int not null auto_increment nameSubjets varchar (45) not null

8.2.DIAGRAMA ENTIDAD-RELACION

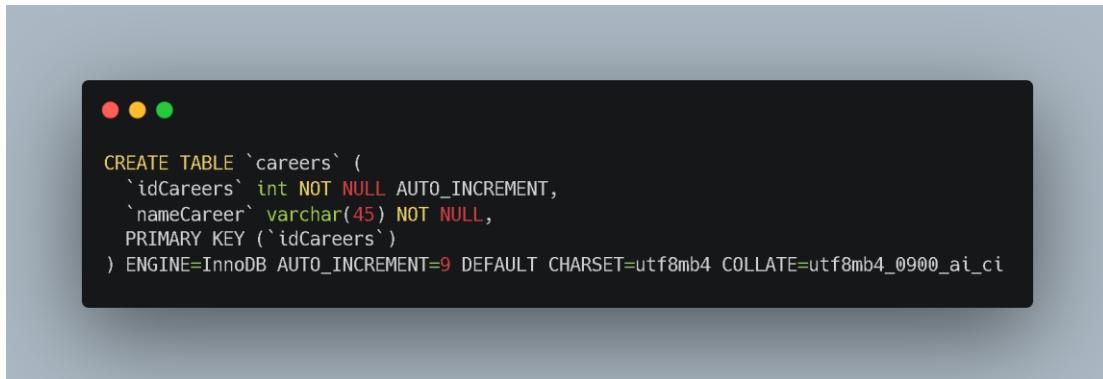


8.3.DIAGRAMA DE ARQUITECTURA



8.4.IMPLEMENTACION DE LA BASE DE DATOS

Tabla carrers



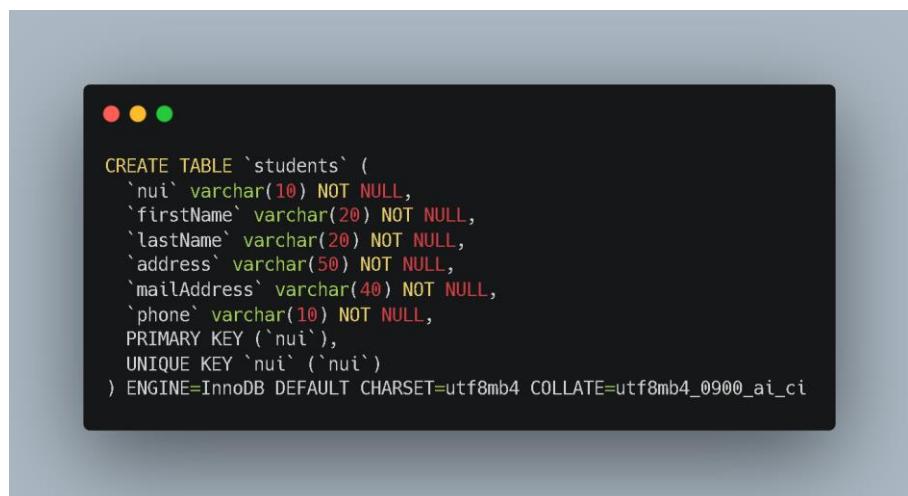
```
CREATE TABLE `careers` (
  `idCareers` int NOT NULL AUTO_INCREMENT,
  `nameCareer` varchar(45) NOT NULL,
  PRIMARY KEY (`idCareers`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

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
```

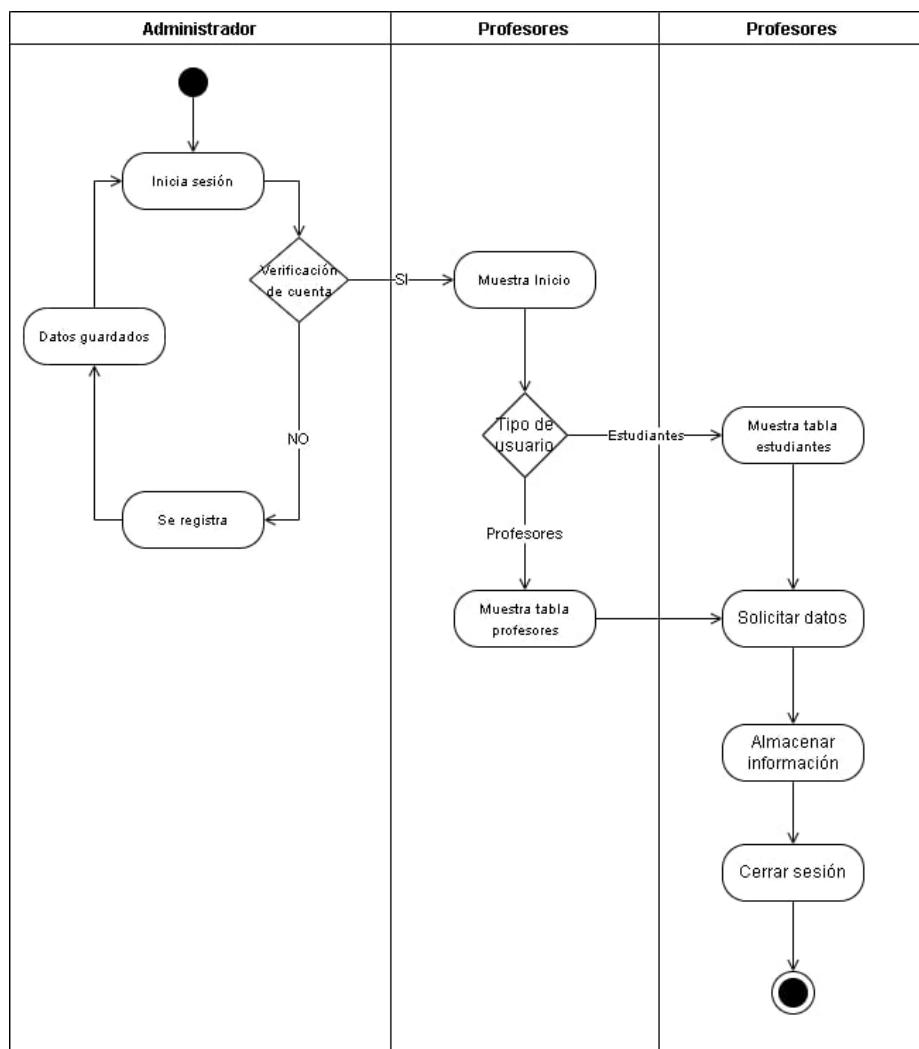
Tabla academicmech

```

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

```

8.5.DIAGRAMA DE ACTIVIDADES



8.6.CODIFICACION DE LA PÁGINA

8.6.1. CÓDIGO HTML EN NETBEANS CON JAVA

8.6.1.1.INTRODUCCIÓN DEL SITIO WEB



The screenshot shows the NetBeans IDE interface with four code panes. The top pane contains the head section of an HTML file, including meta tags and links to Bootstrap and CSS files. The second pane contains the body section, which includes a div for the background and another for the main content with a row and col-md-10 columns. The third pane contains the main content area, starting with several blank lines and then a center block containing a title and a paragraph describing the "College" application. The bottom pane contains a table structure.

```
<%@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>
<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>
<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 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>
</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>
</footer>
</body>
</html>
```

8.6.1.2.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>
```

```
<!-- 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
    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 Frias 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/Navs.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>
```

8.6.1.3. 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>
                    <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>
```

```
<br>
<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 class="col-md-6">
    <main>
        <br>
        <br>
        <br>
        <header>
            <h1 align="center" style="font-family: Roboto,serif; color: #141c27"
                class="animate__animated animate__bounce">
                Asignaturas</h1>
        </header>
```

```

<div class="container" align="center" style="font-family: Roboto">
    <table
        id="tableS"
        data-locale="es-ES"
        data-url="/college/subjects"
        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="idSubjects"><center>ID</center></th>
                <th data-field="nameSubjects"><center>Asignatura</center></th>
                <th data-field="operate" data-formatter="operationsSubjects"
                    data-width="222"><center>Acción</center></th>
            </tr>
        </thead>
    </table>

```

```

        </div>
        </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>
        <%@include file=".libs/ModalsCareers.html"%>
        <%@include file=".libs/ModalsSubjects.html"%>
    </body>
</html>

```

8.6.1.4. 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>

```

```

<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>
</div>
<%@include file="./libs/ModalsStudents.html" %>
</body>
</html>

```

8.6.1.5.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>

```

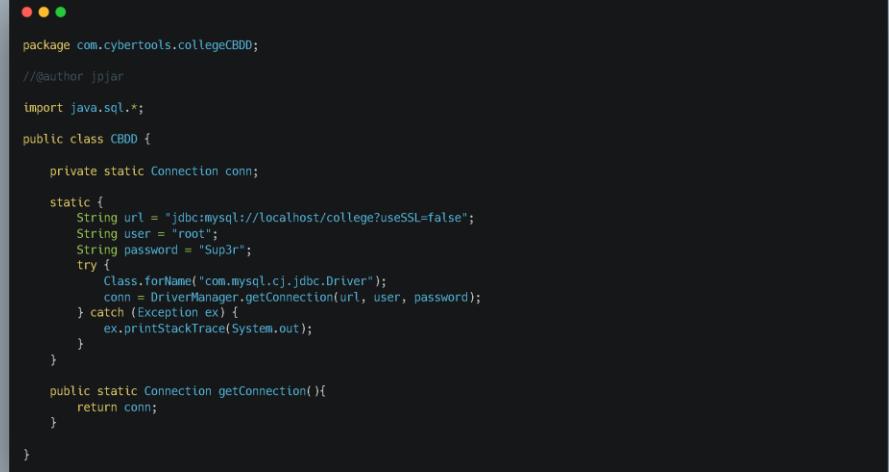


```

<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>
    </footer>
    <%@include file="./libs/ModalsTeachers.html"%>
</body>
</html>

```

8.6.2. CONEXIÓN CON LA BASE DE DATOS



```

package com.cybertools.collegeCBDD;

// @author jpjar

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;
    }
}

```

8.6.3. CONTROLADORES

8.6.3.1. CONTROLADOR DE CARRERAS

```
● ● ●

package com.cybertools.collegeController;

//@author jpjar

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(2, 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;
    }
}
```

8.6.3.2.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);
        }
    }
}
```

8.6.3.3. 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.getNui());
            ResultSet rs = cs.executeQuery();
            while(rs.next()){
                result = rs.getString(1);
            }
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
        return result;
    }
}
```

8.6.3.4. 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;
}
}

```

8.6.3.5. 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;
    }
}
```

8.6.3.6. 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;  
}
```

8.6.4. MODELOS

8.6.4.1.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;
    }
}
```

8.6.4.2. 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;
}

}
```

8.6.4.3.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;
}
}
```

8.6.4.4.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;
    }
}
```

8.6.4.5.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;
}
}

```

8.6.5. SERVLETS

8.6.5.1. 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);
    }
}
}

```

8.6.5.2.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 {

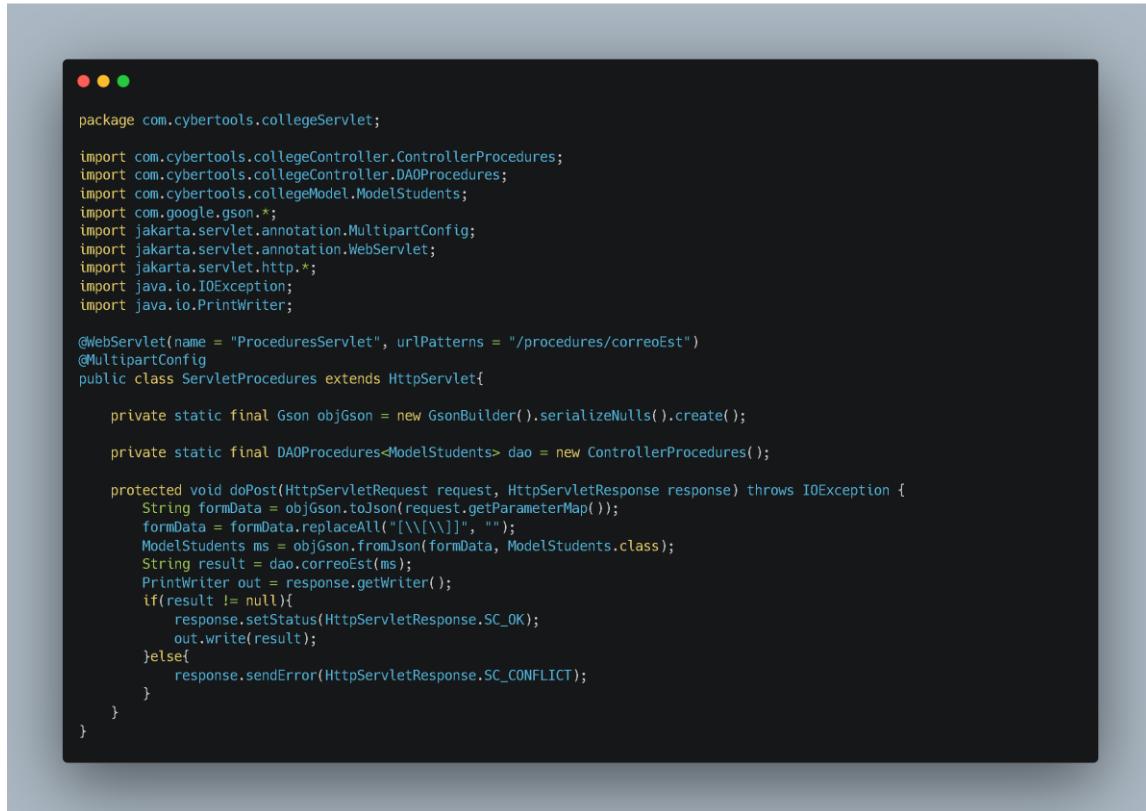
    private static final Gson objGson = new GsonBuilder().serializeNulls().create();
    private static final DAOLogin<ModelUsers> dao = new ControllerLogin();

    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);
        }
    }
}

```

8.6.5.3.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;

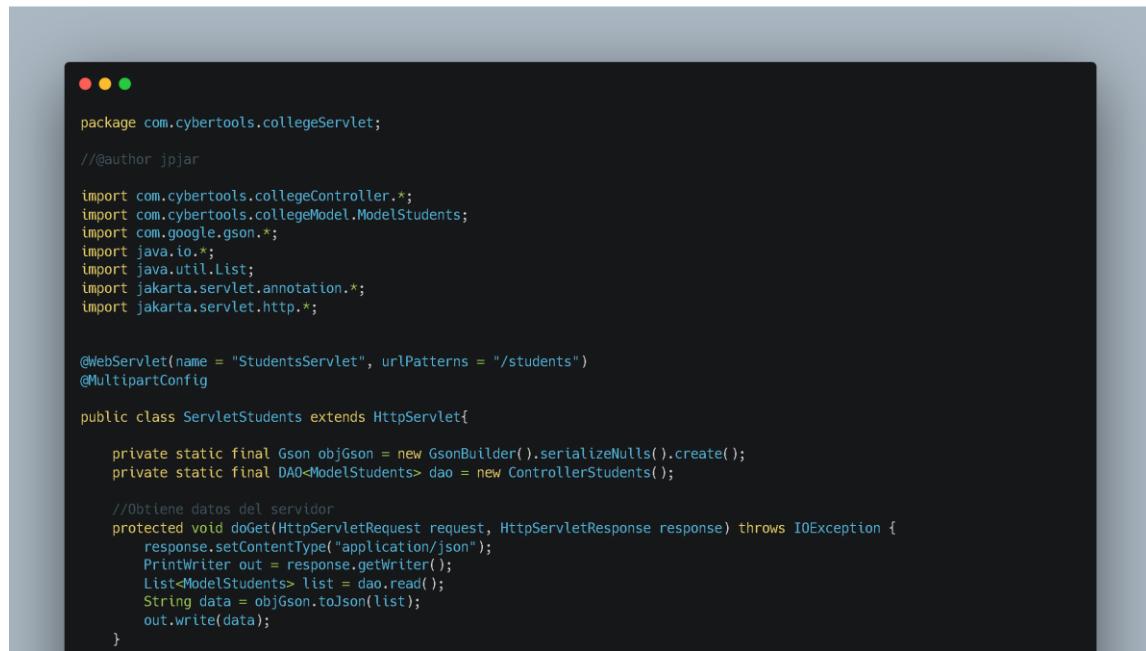
@WebServlet(name = "ProceduresServlet", urlPatterns = "/procedures/correoEst")
@MultipartConfig
public class ServletProcedures extends HttpServlet {

    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("[\\n\\r]", "");
        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);
        }
    }
}
```

8.6.5.4.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);
    }
}
```

```

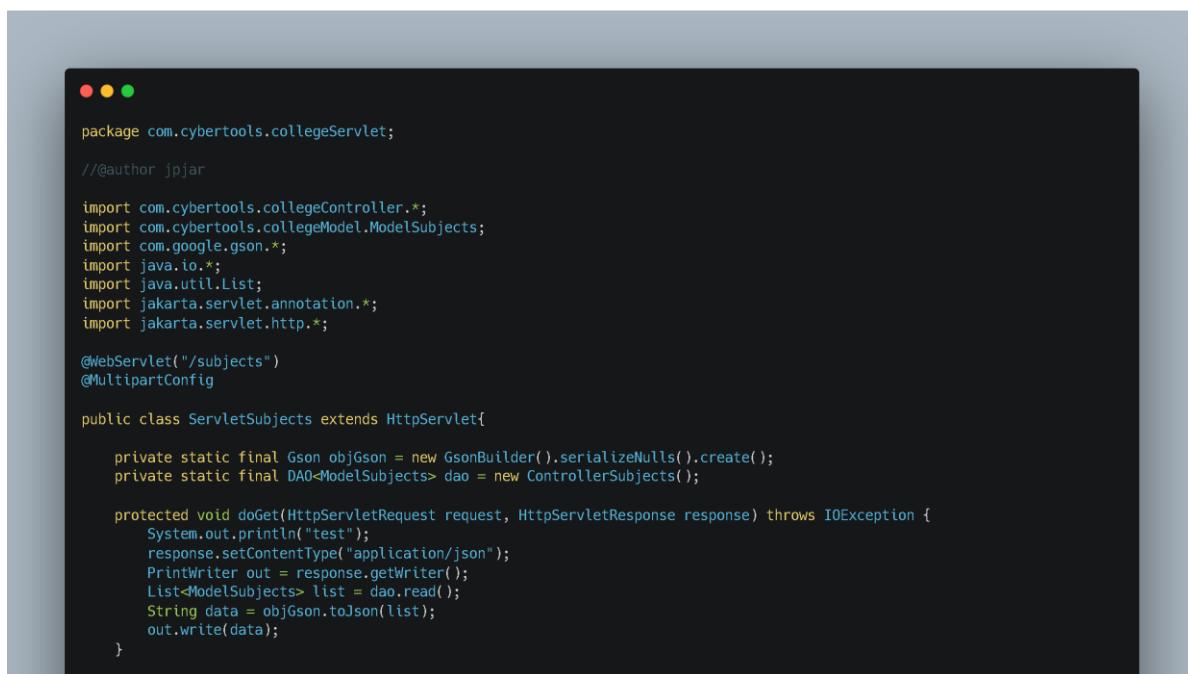
//Crea datos en el server
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException {
    String formData = objGson.toJson(request.getParameterMap()); //optiene todos los datos de una sola vez
    formData = formData.replaceAll("[\\n\\r]", ""); //reemplaza simbolos 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);
    }
}
}

```

8.6.5.5.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);
    }
}
}

```

8.6.5.6.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);
    }
}
}

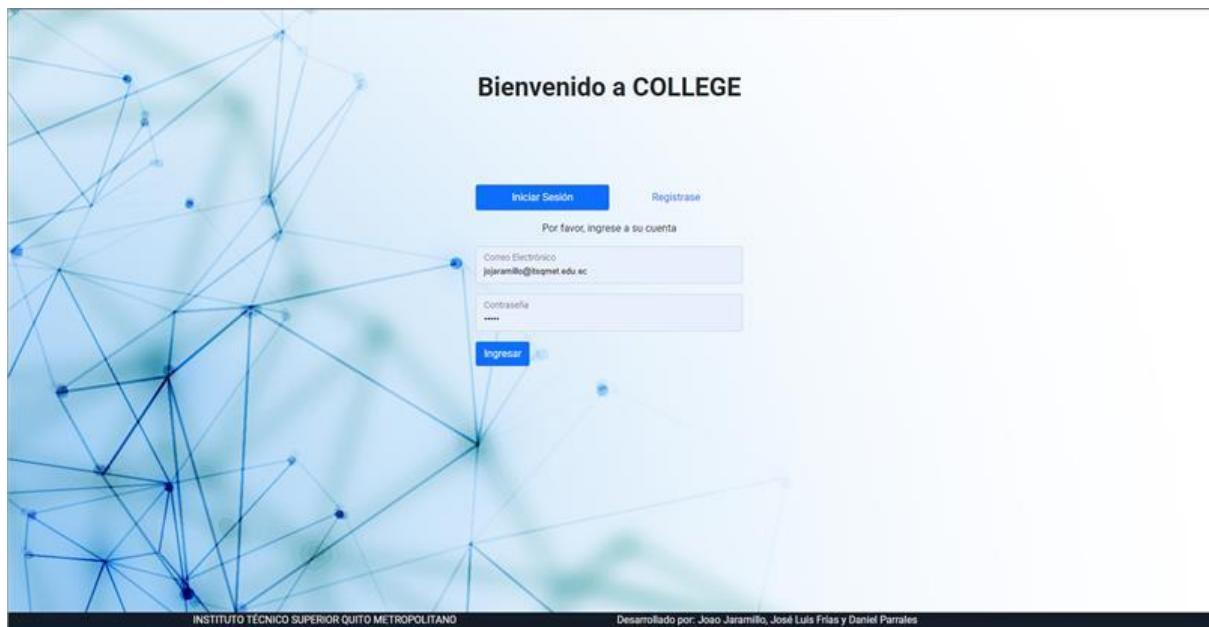
```

8.6.6. DEPENDENCIAS USADAS



9. 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 "Introducción" section of the College application. It includes a brief description of the application's purpose, the "Objetivo General" (General Objective) which is to develop a web application using Java, and the "Objetivos Específicos" (Specific Objectives) which include describing program characteristics, knowing program execution, and implementing a database. 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.

INSTITUTO TÉCNICO SUPERIOR QUITO METROPOLITANO

Desarrollado por: Joao Jaramillo, José Luis Frías y Daniel Parrales.

INSTITUTO TÉCNICO SUPERIOR QUITO METROPOLITANO

Desarrollado por: Joao Jaramillo, José Luis Frías y Daniel Parrales.

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.

10. CONCLUSIONES

Después de lo expuesto podemos concluir que, implementando las librerías correctas, con un código limpio y bien estructurado, pudimos realizar una página web con las funcionalidades que queríamos. Sin olvidar la conexión a la base de datos y a sus respectivas tablas, para un manejo idóneo de la información.

11. RECOMENDACIONES

- Estar a la par de la tecnología actual, de manera que podemos interpretar el código mucho fácil y rápido.
- Distribuir la información HTML en diferentes páginas, para mantener una buena

presentación del sitio web y que sea amigable para el usuario.

- Usar diagramas UML para tener una estructura previa, al desarrollo de la página y la base de datos.

12. BIBLIOGRAFÍA

- Alarcón, J. M. (01 de junio de 2022). *Campus MVP*. Obtenido de <https://www.campusmvp.es/recursos/post/java-que-es-maven-que-es-el-archivo-pom-xml.aspx>
- BLOG, C. (s.f.). *LA INFRAESTRUCTURA TECNOLÓGICA*. Obtenido de CEUPE: <https://www.ceupe.com/blog/infraestructura-tecnologica.html#:~:text=La%20plataforma%20o%20infraestructura%20tecnol%C3%B3gica,gesti%C3%B3n%20de%20los%20equipos%2C%20de>
- Chacón, A. (2003). *Una nueva cara de Internet*.
- Chacon, S., & Straub, B. (2021). *Pro Git*. Apress.
- 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)
- Grupo Conforsa. (2021). *Mi Formación Gratis*. Obtenido de [https://www.miformaciongratis.com/blog-post/html5-que-es-y-para-que-sirve/#:~:text=HTML5%20\(HyperText%20Markup%20Language%2C%20versi%C3%A9n,necesidades%20de%20hoy%20en%20d%C3%A9cada](https://www.miformaciongratis.com/blog-post/html5-que-es-y-para-que-sirve/#:~:text=HTML5%20(HyperText%20Markup%20Language%2C%20versi%C3%A9n,necesidades%20de%20hoy%20en%20d%C3%A9cada).

IBM. (2021). *InfoSphere Data Architect 9.1.2*. IBM, doc. Obtenido de <https://www.ibm.com/docs/es/ida/9.1.2?topic=modeling-logical-data-models>

IBM. (30 de agosto de 2022). *ibm*. Obtenido de <https://www.ibm.com: https://www.ibm.com/docs/es/was-liberty/base?topic=overview-jakarta-java-ee-8-in-liberty>

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

IONOS. (02 de octubre de 2020). *Digital Guide*. Obtenido de <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/bootstrap-5/#:~:text=La%20herramienta%20proporciona%20plantillas%20para,web%20moderno%20resulta%20muy%20sencillo.>

Keep Coding. (5 de abril de 2022). *Home Blog*. Obtenido de [https://keepcoding.io/blog/json-api-que-es-para-que-sirve/#:~:text=JSON%20API%20es%20una%20convenci%C3%B3n%20\(no%20un%20est%C3%A1ndar\)%20para%20expresar,formato%20que%20trabaja%20con%20HTTP](https://keepcoding.io/blog/json-api-que-es-para-que-sirve/#:~:text=JSON%20API%20es%20una%20convenci%C3%B3n%20(no%20un%20est%C3%A1ndar)%20para%20expresar,formato%20que%20trabaja%20con%20HTTP).

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.

Marqués, M. (2009). *Bases de datos*. Universitat Jaume I. Servei de Comunicació i Publicacions.

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

MDN Contributors. (13 de agosto de 2022). *Web Docs.* Obtenido de https://developer.mozilla.org/es/docs/Web/API/Fetch_API

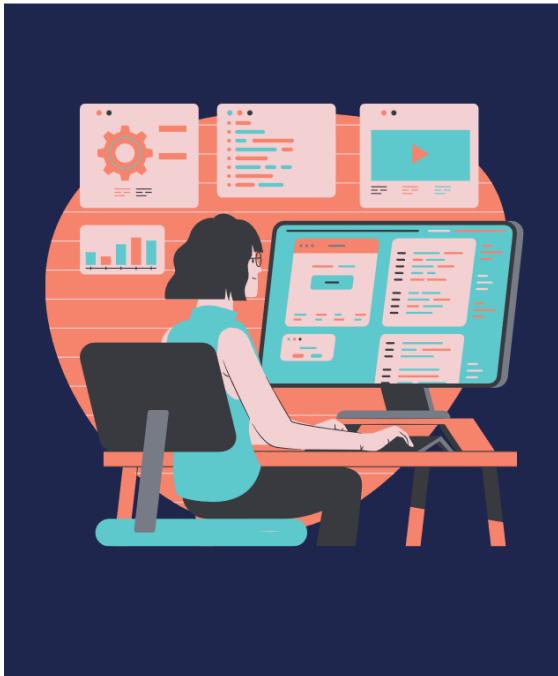
MDN Contributors. (05 de septiembre de 2022). *Web Docs.* Obtenido de <https://developer.mozilla.org/es/docs/Glossary/CSS>

Oracle. (29 de noviembre de 2014). *docs.oracle.* Obtenido de <https://docs.oracle.com/>

oracle. (29 de agosto de 2022). Obtenido de [www.oracle.com:](https://www.oracle.com/mysql/what-is-mysql/) <https://www.oracle.com/mysql/what-is-mysql/>

Sanchez, A. (22 de 10 de 2019). *¿Qué es un parche? De accesorio para tuertos a mejora de programas.* Obtenido de Pandora Fms: <https://pandorafms.com/blog/es/que-es-un-parche/>

13. ANEXOS



Añade el nombre de la
empresa

PROYECTO

Presentación por: Jimmy Daniel Parrales Idrovo
Joao Jaramillo Villalva
José Luis Hernández Frías

Base de datos
Adaptaciones Web
Programación Orientada a Objetos

¿QUÉ ES NUESTRO PROYECTO?



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.



Objetivo General

Desarrollar un Página Web "College" que nos permita la administración de datos para una Institución Educativa mediante la utilización de las diferentes herramientas y los conocimientos adquiridos en las diferentes asignaturas a lo largo de la carrera.

Objetivo Específicos



Desarrollar una interfaz gráfica en HTML intuitiva y amigable para los usuarios.



Adicionar a la página tecnología JSF y funcionalidades a través de JavaScript, que permitan una mejor interacción a los usuarios.



Implementar una base datos para un mejor control de registros, añadiendo diferentes roles y administración de credenciales, también funciones como triggers y procedimientos almacenados.

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.

Sin embargo, muchas veces por falta de recursos no pueden o no ven necesario la implementación de una página web y una base de datos que ayude a la institución a manejar correcta y ordenadamente los datos internos.



JUSTIFICACION

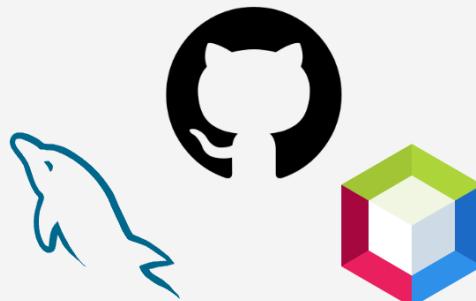


Si queremos que una institución educativa crezca, aparte del buen nivel de enseñanza, debe tener una página web acorde a la calidad de servicio que brinda. Dicha página además de ser amigable a los usuarios, debe ser bien estructurada en todas sus vistas y conectada correctamente a una base de datos que administre la información, genere reportes, procedimientos y funciones en todas sus tablas.



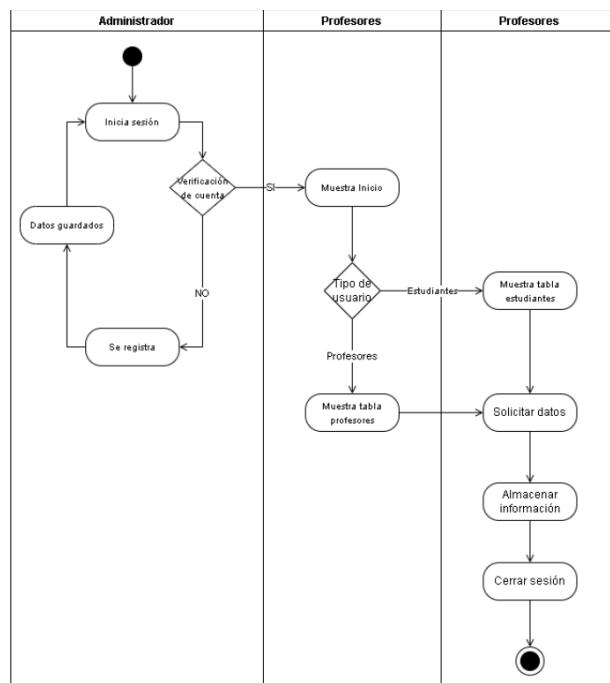
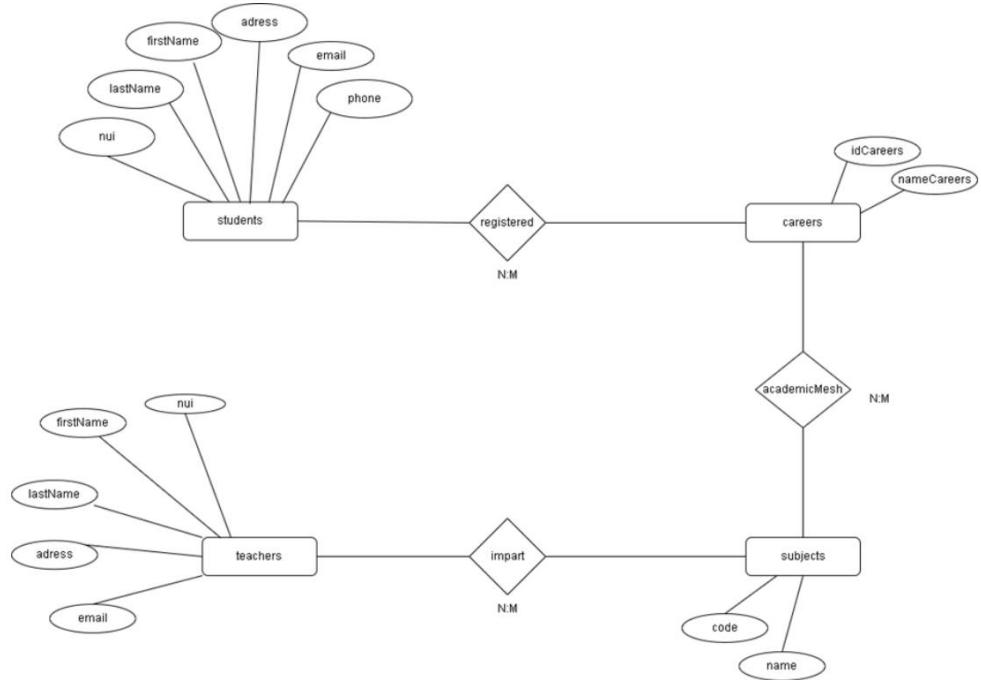
HERRAMIENTAS

Lenguajes:
Java, Jakarta EE y SQL.
Programas:
GitHub, NetBeans, MySQL

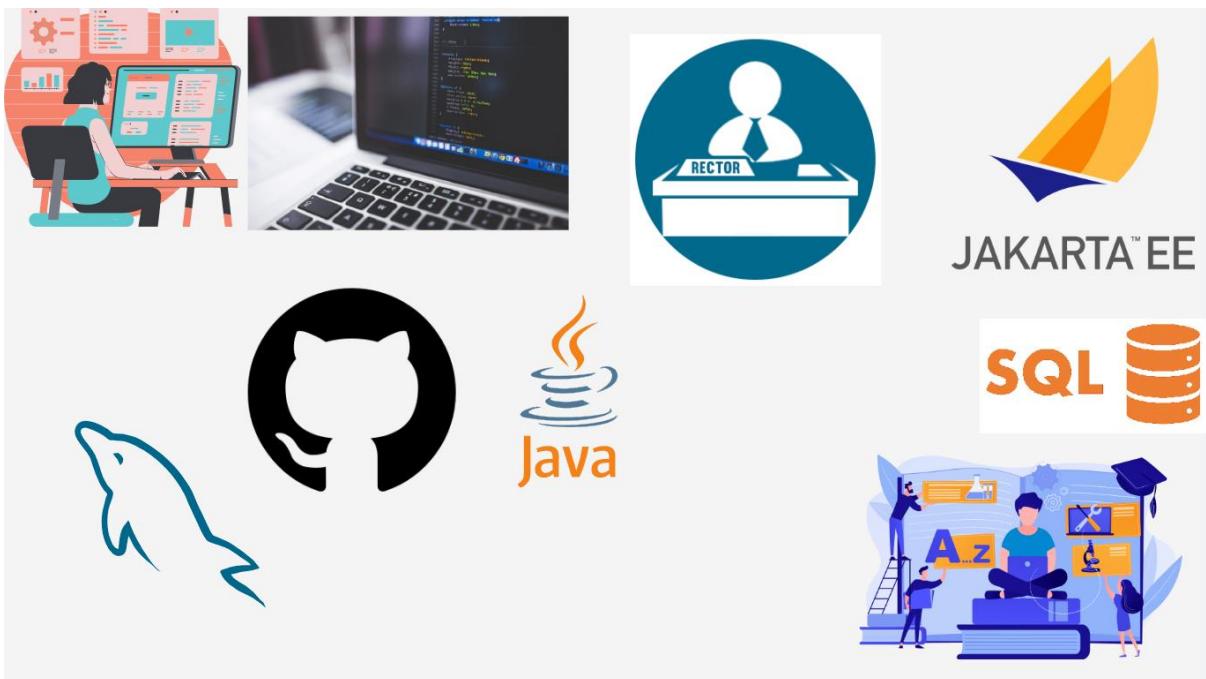


LEVANTAMIENTO Y DIAGRAMAS

Tablas	Atributos
Students	NUI varchar(10) not null unique primary key Prestona [Esc] para salir del modo de edición
Teachers	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
Careers	idCareers int not null auto_increment nameCareer varchar(45) not null
Subjects	idSubjects int not null auto_increment nameSubjects varchar(45) not null



RESULTADOS



Desarrollado en la plataforma de “Canva”

Link de la presentación:

https://www.canva.com/design/DAFK_NzPVns/P2AfY6EG6o8YRn7_tc1T6g/view?utm_content=DAFK_NzPVns&utm_campaign=designshare&utm_medium=link2&utm_source=sh

arebutton