Universidad Michoacana de San Nicolás de Hidalgo

Facultad de Ingeniería Eléctrica

"Diseño e Implementación de un Sistema Web para facilitar el Desarrollo de los Concursos de Oposición dentro de la FIE"

TESIS

Que para obtener el título de:

INGENIERO EN COMPUTACIÓN

Presenta:

Gerardo Lorenzo Cruz

Asesor:

M.C. José Rafael Rodríguez Ochoa

MORELIA, MICHOACÁN. SEPTIEMBRE DEL 2019.

Agradecimientos

- A Dios por cuidarme día a día, y permitir cumplir mis sueños.
- A mis padres Efraín Lorenzo Martínez y Paula Cruz Mateo por el apoyo y cariño incondicional en el transcurso de la carrera.
- A mis hermanos por el apoyo incondicional.
- A mis amigos que prestaron atención y ayuda en el momento que lo necesite.
- Al M.C. Rafael Rodríguez Ochoa por el tiempo y apoyo brindado durante el asesoramiento de la tesis.
- A las instituciones educativas las cuales son parte de mi desarrollo académico, ya que también fueron parte esencial para lograr esta nueva meta.

Dedicatoria

- A Dios por cuidarme y darme sus bendiciones día a día.

- A mis padres por el apoyo en los momentos buenos o malos para concluir este proyecto.

- A mis hermanos por el apoyo que me brindan día a día.

- A mis amigos por la atención y tiempo brindado en tiempos desfavorables.

Índice

Resumen

La presente tesis, muestra el diseño y desarrollo de un sistema web con Python y Django, con él que, sé busca facilitar la evaluación de los expedientes de los profesores que participan en los concursos de oposición, que es un procedimiento selectivo, para determinar que profesor es el idóneo para impartir la materia o cubrir una plaza vacante. Particularmente, el desarrollo se enfoca en los concursos que se realizan en la Facultad de Ingeniería eléctrica de la Universidad Michoacana de San Nicolás de Hidalgo.

Para el sistema web, se consideran diferentes tipos de usuarios, como lo es, un administrador, un integrante de la comisión académica dictaminadora y concursante (posible profesor). Cada persona, tiene acceso al sistema de acuerdo a su tipo de usuario que se registró, para garantizar el buen funcionamiento.

El sistema web, tiene de manera pública, los concursos de oposición, permite registrar a los profesores que tienen la intención de impartir alguna de las materias o bien cubrir alguna plaza vacante dentro de la Facultad de Ingeniería Eléctrica, así como almacenar el documento que el concursante considere adecuado como comprobante de su capacidad académica. Así mismo, los resultados de los concursos, quedan almacenados para llevar un control y en futuros concursos, donde esté involucrada la misma persona no sea complicada su evaluación, ya que se consideraría, sólo lo realizado recientemente por la persona.
Para el desarrollo del sistema web, se utiliza el sistema operativo Linux, ya que es gratuito y optimiza el rendimiento del sistema, además de que soporta múltiples lenguajes de programación; se utiliza el lenguaje de programación de Python, el Framework Django que impulsa el desarrollo de código limpio.

En Django se crea el proyecto que es el sistema web y funcionalidades llamadas aplicaciones, las cuales se agregan en el proyecto, para ello es necesario abrir el archivo `settings.py` que se encuentra dentro del proyecto, en este archivo se hacen las configuraciones del proyecto: se agregan las aplicaciones, configuración de la base de datos, idioma, horario, etc.

El trabajo que se presenta, está formado de 5 capítulos.

El capítulo 1, contiene una introducción al problema a resolver, una propuesta de solución y teoría general del software utilizado como herramienta.

El capítulo 2, hace énfasis en la instalación, configuración y uso de Django.

El capítulo 3, muestra la manera en que se desarrolló el sistema, motivo de este trabajo.

El capítulo 4, se presenta un ejemplo de cómo se debería utilizar el sistema, para los 3 tipos de usuarios considerados en el planteamiento.

Finalmente, el capítulo 5, contiene las conclusiones y trabajos futuros que se pueden agregar al sistema.

Palabras clave

Django, Proyecto, Modelo, Plantilla, Vista.

Abstract

The present thesis shows the design and development of a web system with Python and Django, with which, I want to facilitate the evaluation of the files of the professors who participate in the competitions of opposition, which is a selective procedure, to determine that Professor is the ideal to teach the subject or fill a vacancy. Particularly, the development focuses on the competitions held at the Faculty of Electrical Engineering of the Michoacana University of San Nicolás de Hidalgo.

For the web system, different types of users are considered, as it is, an administrator, a member of the academic committee, and a contestant (possible teacher). Each person has access to the system according to their type of user who registered, to ensure proper functioning.

The web system, publicly, has opposition contests, allows teachers to register who intend to teach any of the subjects or fill a vacancy within the Faculty of Electrical Engineering, as well as store the document that the Contestant considered appropriate as proof of their academic ability. Likewise, the results of the competitions, are stored to keep track and in future competitions, where the same person is involved, its evaluation is not complicated, since it would be considered, only recently performed by the person.

For the development of the web system, the Linux operating system is used, since it is free and optimizes system performance, in addition to supporting multiple programming languages; Python programming language is used, the Django Framework that drives the development of clean code.

Django is composed of a project that is the web system and functionalities called applications, which are added in the project, for this it is necessary to open the settings.py file that is inside the project, in this file the project settings are made : applications, database configuration, language, schedule, etc.

The work presented is made up of 5 chapters.

Chapter 1 contains an introduction to the problem to be solved, a solution proposal and general theory of the software used as a tool.

Chapter 2 emphasizes the installation, configuration and use of Django.

Chapter 3 shows the way in which the system was developed, the reason for this work.

Chapter 4 presents an example of how the system should be used, for the 3 types of users considered in the approach.

Finally, chapter 5 contains the conclusions and future work that can be added to the system.

Keywords

Django, Project, Model, Template, View.

Lista de figuras

Lista de tablas

Glosario de términos

Concurso de oposición	Es un procedimiento selectivo en el que varios profesores participan para impartir alguna materia dentro de la Facultad de Ingeniería eléctrica. Teniendo en cuenta que hay concurso de oposición interno y abierto, tomando en cuenta que estos dependen de un reglamento.
Pip	Es un sistema de gestión de paquetes utilizado para instalar y administrar paquetes de software escritos en Python.

Capítulo 1
Introducción

En éste capítulo, se describe el problema que se presenta al realizar los concursos de oposición dentro de la Facultad de Ingeniería Eléctrica, para seleccionar el profesor idóneo para impartir alguna materia o cubrir alguna plaza vacante.

Se describe y se implementa la solución para que este proceso se realice de manera rápida, sencilla y eficiente.

Se muestra la información a conocer para el desarrollo del proyecto: sistema operativo utilizado, lenguaje de programación “Python”, Framework Django y Bootstrap.

1.1- Introducción al sistema

En la Facultad de Ingeniería Eléctrica, se realiza de manera frecuente, los concursos de oposición, que es un procedimiento selectivo en el que varios profesores participan, para determinar cuál es el idóneo para impartir cada materia vacante, o bien, cubrir una plaza disponible.

Durante el proceso del concurso, se realizan varias tareas, entre las que se destacan las siguientes:

1) Revisión de currículum vitae.
2) Examen oral de un tema.
3) Revisión de un tema por escrito.
4) Y demás requisitos indicados en la convocatoria.

Para ello, se implementa un sistema web que permite la captura de información referente al concurso. La información, ésta dividida en 3 tipos diferentes de usuario según el grado de responsabilidad. Estos son:

1) Administrador.
2) Comisión académica.
3) Concursante.

Cabe mencionar, que la decisión de otorgar o no, una materia o plaza vacante, es criterio de los miembros de la comisión académica y no del sistema que aquí se implementa, ya que el sistema, solamente es una herramienta para facilitar el manejo de la información.

1.2- Problema a resolver

Una de las tareas del concurso de oposición, consiste en revisar la documentación del currículum que presenta el profesor concursante. Como resultado de ésta revisión, se llena un formato, en el que se asigna un puntaje de acuerdo a lo indicado en las tablas de valoración. Una tabla de valoración, consiste en la especificación de la documentación que se pueda presentar en un examen de oposición, así como la asignación de puntaje.

Al revisar un expediente, normalmente es complicado, debido al volumen de documentos que entrega cada uno de los profesores participantes en el concurso. De tal manera, que sí, ese mismo profesor participa en algún otro concurso, habría que revisar nuevamente el mismo expediente que se presentó con anterioridad.

Incluso, con el probable riesgo de recibir un puntaje diferente debido a que no siempre es la misma comisión académica responsable del concurso interno.

1.3- Solución propuesta

Se plantea implementar un sistema web en el que se evite, en lo posible, el tener que revisar varias veces el mismo documento.

Para ello, el documento se presentaría solo una vez, ya que se almacena digitalmente, así como el puntaje, que, a criterio de la comisión académica en turno, se le asigne.

En el caso de que un profesor, que participa en un concurso, vuelva a participar, solamente lo indicara en su currículum sin incluir el documento que lo avale. Lo anterior porque ya se tiene en el sistema.

Queda a criterio de la comisión académica en turno, así como sucede actualmente, conservar el puntaje asignado o bien modificarlo.

Se considera que, si un profesor concursante, fuese autor de un libro, solamente se almacena la portada dentro del sistema.

Se considera, que la primera vez que se presente el profesor al concurso, debe presentar en físico su documentación para su cotejo.

Así, mismo queda a responsabilidad del profesor concursante, subir al sistema, los archivos correspondientes a la documentación.

1.4- Objetivo general

Proporcionar un sistema web que permita almacenar, de manera digital, los documentos que avalen el currículum que presenta cada profesor concursante. Así, mismo, que almacene los puntajes asignados a cada uno de ellos en un concurso de oposición.

1.5- Objetivos particulares

Crear una página web base, que sirva de referencia para las ampliaciones futuras. La página web que se presenta, cubre solamente la primera parte de lo que se pretende que tenga. Es decir, solamente almacena la documentación que avala a los currículums, así como el puntaje del concurso en que participa cada profesor.

1.6- Justificación

Cuando se revisa un expediente, la comisión académica en turno, debe comprobar la documentación presentada por el profesor concursante. Así mismo, la comisión asignara un puntaje que considere adecuado, según la tabla de valoración respectiva.
Suponiendo, que un profesor participara en un concurso futuro, este deberá presentar nuevamente su expediente y con ello, la comisión revisara, de nueva cuenta, el mismo expediente anterior incluyendo las actualizaciones al currículum, si es el caso.

Es posible, que, en la misma evaluación del expediente, y con la misma comisión académica, se asigne un puntaje diferente al anterior. Esto es debido a que la asignación de puntos, depende de la apreciación, en ese momento, de la comisión académica.

Finalmente, cabe la posibilidad, de revisar solamente el trabajo de fechas recientes del Profesor concursante, dejando, a criterio de la comisión, revisar o no lo que se revisó en algún concurso anterior.

1.7- Información necesaria para desarrollar el proyecto

En éste apartado, se menciona la información necesaria para desarrollar el proyecto, así como se describe el sistema operativo utilizado.

1.7.1- Sistema operativo para el desarrollo del proyecto

El sistema operativo utilizado para el desarrollo del proyecto es Linux, ya que es el sistema operativo utilizado en la Facultad de Ingeniería Eléctrica, así como también permite desarrollar sistemas web, ya que la mayoría de los servidores son Linux. Todos los servicios que se encuentran en Internet funcionan de forma nativa y no son complicados de instalar, incluso actualmente hay asistentes que lo hacen de forma bastante automatizada[1].

Ventajas de usar Linux:

a) Gestión cómoda de instalación de software o servicios para desarrollar el proyecto.
b) Magnífico rendimiento.

Desventajas de usar Linux:

a) Mayor tiempo para instalar servicios o realizar configuraciones en el sistema Linux por medio de la terminal [2].

1.7.2- Dominio

Un dominio, es el nombre único y exclusivo que se le da a un sitio web en Internet, para que cualquier persona y desde cualquier lugar pueda acceder al sitio web[4].

1.7.3- Hosting (Alojamiento web)

El hosting, es tener espacio en el disco duro de un servidor, que está preparado con los programas necesarios para asociar un dominio, para poder subir un sitio web[3].

1.7.4- Python

Python es un lenguaje de programación de alto nivel, interpretado para programación de propósito general, tiene una filosofía de diseño, que enfatiza la legibilidad del código, notablemente utilizando espacios en blanco significativos, proporciona construcciones que permiten una programación clara en escalas pequeñas y grandes, ya que presenta un sistema de tipo dinámico y administración de memoria automática; también admite múltiples paradigmas de programación, incluidos los orientados a objetos, imperativos, funcionales y de procedimiento, y tiene una biblioteca estándar amplia y completa[5].

1.7.5- Django

Django es un Framework web Python de alto nivel, que fomenta el desarrollo rápido, un diseño limpio y pragmático; se encarga gran parte de las complicaciones del desarrollo web, fue diseñado para ayudar a los desarrolladores a llevar las aplicaciones desde el concepto hasta su finalización lo más rápido posible, tomando en serio la seguridad y ayuda a los desarrolladores a evitar muchos errores comunes de seguridad; también es extremadamente escalable, ya que algunos de los sitios más ocupados de la Web aprovechan la capacidad de Django para escalar de forma rápida y flexible[6].

1.7.6- Entornos virtuales en Python con Virtualenv

Un entorno virtual, es un ambiente creado con el objetivo de aislar recursos como librerías, programas o módulos necesarios que se necesitan instalar para el funcionamiento del sistema web[7].

Python brinda “Virtualenv”, un programa que permite crear entornos virtuales de Python.

1.7.7- Bootstrap

Bootstrap es un Framework para diseñar sitios y aplicaciones web. Contiene plantillas de diseño basadas en HTML y CSS para tipografía, formularios, botones, navegación y otros componentes de la interfaz, así como extensiones de Java Script opcionales[8].

1.7.8- jQuery

jQuery es una biblioteca de Java Script rápida, pequeña y rica en funciones. Hace cosas como el recorrido y manipulación de documentos HTML, manejo de eventos, animación, y Ajax mucho más simple con una API fácil de usar que funciona en una multitud de navegadores. Con una combinación de versatilidad y extensibilidad, jQuery ha cambiado la forma en que millones de personas escriben Java Script[9].

1.7.9- Base de datos

La palabra “datos” hace referencia a hechos conocidos que pueden registrarse: números telefónicos, direcciones, nombres, etc [10].

La base de datos, es un conjunto de datos estructurados y almacenados de forma sistemática, con el objetivo de facilitar su utilización posteriormente, como poder almacenar, gestionar, actualizar y acceder a los datos fácilmente.

1.7.9.1- Sistemas gestores de base de datos

Los programas denominados Sistemas Gestores de Base de Datos, abreviados SGDB, en ingles Data Base Management System (DBMS), permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada.

Existen diferentes gestores de base de datos como lo es Oracle, MySQL, PostGresSQL, etc.

Se utiliza MySQL, ya que es un sistema de gestión de bases de datos relacional de código abierto (RDBMS) pero sobre todo gratuito.

MYSQL es un sistema multiplataforma, escalable, donde las transacciones son atómicas y consistentes. MYSQL toma en cuenta la velocidad al realizar las operaciones, pero sobre todo lo más importante, la baja probabilidad de corromper datos, incluso si los errores no se producen en el propio gestor, si no en el sistema en el que está instalado.

En el trabajo que se presenta, no se utiliza una base de datos como tal. Es decir, se hace uso de tablas, pero como una herramienta de apoyo y no como un sistema de almacenamiento de información que amerite el uso del potencial de un gestor de base de datos.

Capítulo 2
Manual de configuración y uso de Django

En éste capítulo, se describe el origen y ventajas de Django para el desarrollo web, así como la forma de instalar Django dentro de los entornos virtuales.

Se describe la estructura y configuración de los archivos que se generan al crear el proyecto y las aplicaciones que lo conforman.

Se explica las partes y el funcionamiento de cada archivo utilizado para entender el funcionamiento del modelo MTV de Django.

2.1- Origen de Django

Django fue construido como una herramienta para resolver problemas reales en un entorno empresarial. Así como para optimizar el tiempo de desarrollo y los requerimientos exigentes de los desarrolladores web.

2.2- Ventajas de Django

Django impulsa el desarrollo de código limpio al promover buenas prácticas de desarrollo web. Django, sigue el principio DRY (Don't repeat yourself - no te repitas), usando una modificación de la arquitectura MVC (Modelo-Vista-Controlador), llamada MTV (Model-Template–View), está forma de trabajar permite que sea pragmático.

2.3- Instalación de Django en Linux con pip desde terminal

Para instalar Django con pip, es necesario saber que pip es un sistema de gestión de paquetes utilizado para instalar y administrar paquetes de software escritos en Python.

Procedimiento para instalar Django:

1) `sudo apt-get install python3-pip`
2) La línea de código anterior, crea una nueva carpeta llamada pip3, para hacer un uso de la herramienta pip más ágil, estableceremos un enlace hacia pip:

   ```
   which pip3

   ln –s /usr/bin/pip3 /usr/bin/pip
   ```

3) Para comprobar la versión de pip instalada se teclea el comando:
   `pip –V`

4) Una vez instalado pip, se puede usar para instalar Django:

```
pip3 install django==1.9
```

Con 'django==1.9' se especifica qué versión se desea obtener, por lo que si se desea alguna versión diferente, sólo se modifica el número de versión (por ejemplo, django==1.7; django==1.8…)

5) Revisar la versión instalada con el comando:

```
django-admin --versión
```

6) Para comprobar que se instalado correctamente mediante un pequeño script incluido con las dependencias de Django:

```
python3
import django
print(django.get_version())
```

2.4- Virtualenv

Permite crear entornos útiles, cuando en Python se desea aislar un entorno con unas librerías determinadas de otro, por lo que se utiliza el siguiente comando para la instalación.

```
pip install virtualenv
```

Al igual que con Django en el caso de pip, aquí también se puede seleccionar qué versión de Python se desea descargar e instalar, con el comando que se menciona, pudiendo aplicar diferentes versiones a distintos entornos virtuales:

```
virtualenv --python=python3.4 nombre_entorno_virtual
```

El comando mencionado anteriormente, crea una nueva carpeta llamada "nombre_entorno_virtual", que contiene los directorios bin y lib. Ahora se puede activar el entorno virtual para poder instalar las herramientas deseadas, en nuestro caso Django, para empezar, se activa el entorno virtual con el comando siguiente:

```
Source nombre_entorno/bin/actívate
```

```
nombre_entorno/bin/actívate
```
#usar este comando en caso de no tener el comando `source`.

Dentro del entorno virtual descargar e instalar Django:

```
pip install django==1.9
```

Para comprobar que todo está correcto, se usa el comando que permite ver la versión:

```
django-admin --versión
```

2.5- Crear proyecto con Django

Teniendo en cuenta que el nombre del proyecto es la aplicación web, se abre una terminal, se coloca en la carpeta donde se desea crear el proyecto y escribir los comandos siguientes para preparar el entorno de trabajo:

`Virtualenv -p python3 env` (env→ Nombre del entorno virtual).

`source env/bin/actívate` → Activar entorno virtual.

`pip install django` → Instalar Django en el entorno virtual.

`Deactivate` → Desactiva entorno virtual.

2.5.1- Comandos para crear proyecto

El comando para crear el proyecto es el que se menciona en la línea siguiente:

```
django-admin start project "nombre_del_proyecto"
```

Esta instrucción crea un archivo llamado manage.py y un directorio con el nombre del proyecto, teniendo dentro 4 archivos distribuidos de la siguiente manera:

- `manage.py`
- `nombre_del_proyecto`
  - `__init__.py`
  - `settings.py`
  - `urls.py`
  - `wsgi.py`

Para verificar que el proyecto está funcionando correctamente en la terminal se escribe:

```
Python manage.py runserver
```

Obteniendo la Figura 2.1.

Figura 2.1- Servidor corriendo

Abrir en el navegador web, la dirección `http://127.0.0.1:8000`, obteniendo la Figura 2.2.

Figura 2.2- Proyecto en Django creado correctamente

Hasta el momento ya se tiene el proyecto creado; en el caso de que salga un error, porque el puerto asignado está en uso, como se observa en la Figura 2.3.

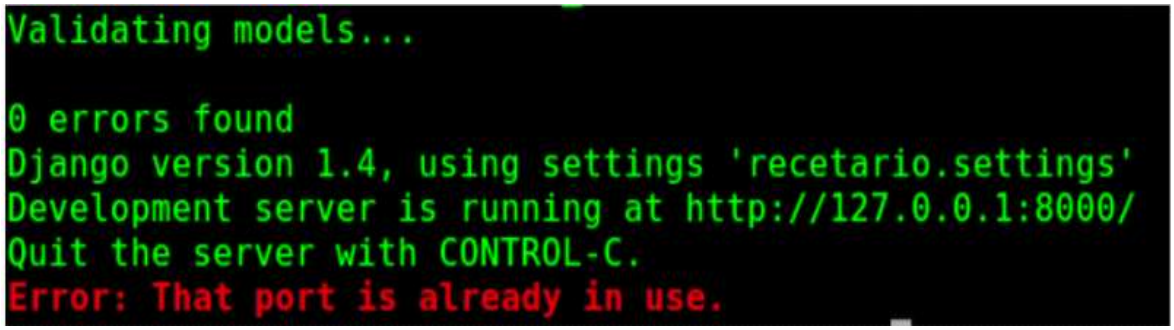


Figura 2.3- Puerto ocupado

Observando la Figura 2.3, muestra un error, ya que el puerto se encuentra ocupado, por lo que es necesario indicar qué puerto usaremos para lanzar nuevamente el servicio, por ejemplo, si se desea usar el puerto 8888, entonces colocar dicho puerto a utilizar como se muestra en la línea siguiente:

```
python manage.py runserver 8888
```

Se usa la dirección `http://127.0.0.1:8888`, para verificar que el puerto no esté ocupado obteniendo la Figura 2.4.

Figura 2.4- Nuevo puerto desocupado

2.5.2- Comandos para crear aplicación

Cada proyecto necesita de módulos o funcionalidades, las cuales en Django se conocen como aplicaciones.

Un proyecto puede tener muchas aplicaciones, para crear las aplicaciones en la carpeta del proyecto, se escribir la siguiente línea de código en la terminal:

```
python manage.py startapp "nombre_aplicación"
```

La línea de código mencionada anteriormente, crea un directorio, y dentro de esté cuatro archivos más, por lo que se obtiene una estructura de archivos como se menciona a continuación:

- `manage.py`
- `nombre_del_proyecto`
  - `__init__.py`
  - `settings.py`
  - `urls.py`
  - `wsgi.py`
- `nombre_aplicación`
  - `__init__.py`
  - `models.py`
  - `test.py`
  - `views.py`

2.5.3- Archivo `settings.py`

El archivo `settings.py`, es una parte muy importante del proyecto, ya que permite configurar la conexión a la base de datos, la zona horaria, el idioma, los directorios principales del proyecto y las aplicaciones del proyecto, etc.

2.5.4- Instrucciones principales a configurar

2.5.4.1- Codificación de caracteres

El idioma que se utiliza está lleno de caracteres especiales, los más comunes las tildes y las eñes, para manejar esto eficientemente en Django, es necesario agregar la siguiente línea al archivo `settings.py`:

```
#encoding:utf-8
```

2.5.4.2- Ruta del proyecto

La configuración de la ruta para el proyecto, permite lanzar la aplicación desde cualquier directorio y mover el proyecto a cualquier computador con Django instalado. Para ello, es necesario escribir las siguientes líneas de código en el archivo `settings.py`:

```
# Identificando la ruta del proyecto
import os
RUTA_PROYECTO = os.path.dirname(os.path.realpath(__file__))
```

NOTA:

> Si no se configura la ruta del proyecto, cada vez que se cambia de directorio o de PC, se tendrá que cambiar las rutas de las plantillas, archivos estáticos y directorio de subida de contenido de los usuarios.

2.5.4.3- Configuración de la base de datos

Es necesario configurar la conexión a la base de datos, según las necesidades que se tangan, Django soporta de manera predeterminada la conexión con postgresql, MySQL, sqlite3. Para configurar la conexión a la base de datos deseada, buscar la sección de DATABASES en el archivo `settings.py` y dejarla de la siguiente manera:

```
DATABASES = {
'default': {
'ENGINE':'django.db.backends.sqlite3',
    'NAME': 'recetario.db',
    'USER': '', # Not used with sqlite3.
    'PASSWORD': '', # Not used with sqlite3.
    'HOST': '',
    'PORT': '',}}
```

2.5.4.4- Zona horaria

Cuando el soporte para zonas horarias está habilitado, Django almacena información de fecha y hora en UTC (Universal Time Coordinated – Tiempo Coordinado Universal) en la base de datos, utiliza objetos de fecha y hora reconocidos por la zona horaria internamente, y los traduce a la zona horaria del usuario final en platillas y formularios. Esto es útil si los usuarios viven en más de una zona horaria y desea mostrar la fecha y hora de acuerdo con el reloj de cada usuario.

En éste caso se configuró a la zona horaria UTC.

```
TIME_ZONE = 'UTC'
```

2.5.4.5- Configuración del idioma

Django permite configurar el idioma que se usará de manera predeterminada, para su funcionamiento, para configurarlo se debe buscar la siguiente línea:

```
LANGUAGE_CODE = 'en-us'
```

Una vez ubicada la línea, modificarla como se menciona a continuación, ya que se desea tener el idioma en español de México:

```
LANGUAGE_CODE = 'es-mex'
```

2.5.4.6- Aplicaciones instaladas

Un proyecto en Django necesita de aplicaciones, algunas ya vienen configuradas de manera predeterminada, pero para habilitar las aplicaciones creadas por el desarrollador, se debe buscar la siguiente sección, que se encuentra en el archivo `settings.py` y agregar cada aplicación creada como se muestra continuación.

```
INSTALLED_APPS= (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.admin',
    'django.contrib.admindocs',
    'nombre_aplicación',)
```

2.5.4.7- Creación de la base de datos

Al momento no se ha creado la base de datos o las tablas predeterminadas del proyecto, solo se han configurado los parámetros de conexión; Para crear la base de datos, se debe colocar desde la terminal las siguientes instrucciones (recordando que todo se realiza en la carpeta de proyecto):

- `python manage.py makemigrations`
- `python manage.py migrate`

2.5.5- Entendiendo como trabaja Django

Funcionamiento del patrón MVC:

Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control, en tres componentes distintos.

El Modelo del patrón MVC, contiene una representación de los datos que maneja el sistema, su lógica de negocio y sus mecanismos de persistencia.

La Vista del patrón MVC o interfaz de usuario, compone la información que se envía al cliente y los mecanismos de interacción con éste.

El Controlador del patrón MVC, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

Django es un Framework que trabaja con el patrón MTV (una modificación del patrón MVC), esto se debe a que los desarrolladores no tuvieron la intención de seguir algún patrón de desarrollo, sino hacerlo lo más funcional posible.

Para empezar a entender el patrón MTV con el que trabaja Django se debe fijar la relación con el patrón MVC, como se muestra en la Tabla 2.1.

Tabla 2.1- Relación entre MVC Y MTV

MVC	DJANGO (MTV)
Modelo	Modelo
Vista	Témplate (Plantilla)
Controlador	Vista

Para entender mejor el funcionamiento de MTV de Django, se observar en la Figura 2.5, que el navegador manda una solicitud, de acuerdo a la URL que se coloca, esta busca la vista que le corresponda e interactúa con el modelo para obtener los datos correspondientes, una vez obtenidos los datos, la vista llama a la Template (Plantilla) y está rende riza la respuesta a la solicitud del navegador.

Figura 2.5- Funcionamiento del MTV de Django

2.5.5.1- El modelo en Django

Un modelo, es la fuente única y definitiva de información sobre sus datos.

Contiene los campos y comportamientos esenciales de los datos que está almacenando. En general, cada modelo se asigna a una única tabla de base de datos, donde cada modelo es una clase de Python de subclases `django.db.models.Model`, también posee métodos o funciones los cuales permiten asignar valores a los campos del modelo.

El código del modelo de ejemplo de la Figura 2.6, define una persona que tiene nombre, apellido, fecha de registro, así como un método llamado Registro:

```
from django.db import models
from django.utils import timezone
class Persona(models.Model):
  Nombre   = models.CharField(max_length=30)
  Apellido = models.CharField(max_length=30)
  Registro_Fecha=models.DateTimeField(blank=True,null=True)
  #metodo para asignar la fecha a el campo Registro_Fecha
    def Registro(self):
      self. Registro_Fecha = timezone.now()
      self.save()
```

Figura 2.6- Código para generar el modelo de una persona

El Nombre, Apellido y Registro Fecha son campos del modelo. Cada campo se especifica como un atributo de clase y cada atributo se asigna a una columna de base de datos.

2.5.5.2- La vista en Django

Una función vista es simplemente una función de Python que toma una solicitud web y devuelve una respuesta web, esto quiere decir que su propósito es determinar qué datos serán visualizados.

La vista también se encarga de tareas conocidas como, la autenticación con servicios externos y la validación de datos a través de formularios. Lo más importante a entender con respecto a la vista, es que no tiene nada que ver con el estilo de presentación de los datos, sólo se encarga de los datos, la presentación es tarea de la plantilla.

El código de la vista de la Figura 2.7, devuelve la fecha y hora actual, como un documento HTML:

```
from django.http import HttpRespose
import datetime
def fecha_hora(request):
   actual = datetime.now()
   html = "<html><body>Fecha y hora actual%s.</body></html>" %
actual
   return HttpResponse(html)
```

Figura 2.7- Código para generar una vista de fecha_hora

2.5.5.3- La plantilla en Django

La plantilla, es básicamente una página HTML con algunas etiquetas extras propias de Django, que recibe los datos de la vista y luego los organiza para la presentación al navegador web.

Las etiquetas que Django usa para las plantillas, permiten que sea flexible para los diseñadores, incluso tiene estructuras de control de datos como `if`, por si es necesaria una presentación lógica de los datos. Esto permite que la lógica del sistema siga permaneciendo en la vista.

Por ejemplo, al observar el código de la plantilla de la Figura 2.8, nuestra en el navegador un texto "My plantilla" y los datos que se reciben de la vista, los cuales se visualizan con la etiqueta `for`.

En el código se observan las etiquetas propias de Django, las cuales van dentro de {%block nombre %} serrándola con un `{%endblock%}` o la etiqueta `for` que de igual va dentro de `{% for i in datos %}` operaciones `{endfor}`.

```
{%  extends 'core/base.html' %}
{% load static %}
{% block title %} Ejemplo de Template {% endblock%}
{% block headers%}
<h1>My plantilla</h1>
{% endblock %}
{% block content%}
    {% for i in datos%}
        Datos : {{i.dato}}
    {% endfor%}
{% endblock%}
```

Figura 2.8- Código de una plantilla que solo muestra la estructura y algunas etiquetas de Django

2.5.5.4- Configuración de las rutas

Django posee un mapeo de URLS las cuales están construido con expresiones regulares en Python, que permite controlar el despliegue de las vistas. Esta configuración, es conocida como URLConf.

En la Figura 2.9, se observa el funcionamiento del modelo MTV, así como el funcionamiento de las URLConf; Primero se lee la URL que el usuario solicitó, se busca la vista apropiada para la solicitud, la vista realiza las operaciones necesarias con el modelo, posteriormente llama a la plantilla y esta rende riza la respuesta a la solicitud del navegador.

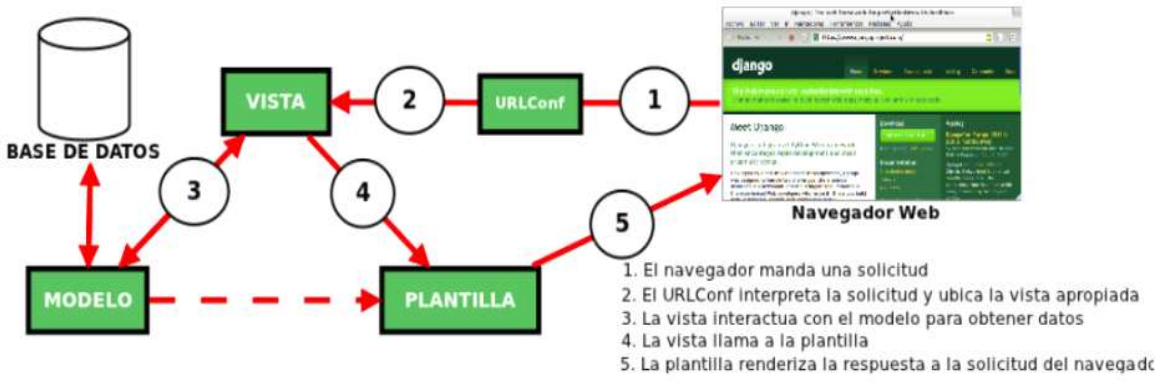


Figura 2.9- Funcionamiento del MTV de Django y su URLConf

2.5.5.5- Archivos predeterminados

Los archivos predeterminados son los que se crean de manera predeterminada, como los que se mencionan:

2.5.5.5.1- Archivos del proyecto

`__init__.py:`	Es un archivo vacío que le dice a Python que debe considerar este directorio como un paquete de Python.
`manage.py:`	Contiene una porción de código que permite interactuar con el proyecto de Django de muchas formas.
`settings.py:`	Contiene todas las configuraciones para el proyecto.
`urls.py:`	Contiene las rutas que están disponibles en el proyecto, manejado por `URLConf`.

2.5.5.5.2- Archivos de la aplicación

`__init__.py:`	Es un archivo vacío que le dice a Python que debe considerar este directorio como un paquete de Python
`models.py:`	Se declaran las clases del modelo.
`views.py:`	Se declaran las funciones de la vista.
`test.py:`	Se declaran las pruebas necesarias para la aplicación.

2.5.5.6- El modelo de datos

Una vez que se tiene instalado Django, y ya se creó el proyecto es necesario definir el modelo de datos para la aplicación.

2.5.5.6.1- El modelo

Un modelo, es la representación de los datos de la aplicación; Contiene los campos básicos y el comportamiento de los datos que serán almacenados; Por lo general, cada modelo se convierte en una tabla de la base de datos, por lo que es necesario considerar lo siguiente:

- ✓ Cada modelo es una subclase de `django.db.models.Model`.
- ✓ Cada atributo de un modelo representa a un campo de una tabla.
- ✓ Django automáticamente da acceso a la base de datos.

2.5.5.7- El Shell de Django

Los modelos permiten manipular los datos como lo es: registrar, editar, actualizar consultar, eliminar y realizar operaciones o funciones con ellos; Toda esta manipulación de datos se realiza en las vistas y posteriormente en las plantillas para mostrar los resultados en el navegador, esta manipulación se le conoce generalmente como consultas.

El Shell, es el intérprete interactivo de Python, que permite probar los modelos, hacer consultas, analizar resultados, antes de elaborar las vistas. Es muy útil, si se desea ahorrar tiempo al momento de responder a los requerimientos, que los usuarios de la aplicación puedan necesitar. Para acceder al Shell, se abre una terminal o ventana de comandos, ubicarse en la carpeta del proyecto (en donde se encuentre el archivo manage.py) y escribir el comando: `python manage.py shell`.

2.5.5.7.1- Las consultas

Las consultas en base a los modelos de Django, son la base de todo el desarrollo en este Framework, estas consultas permiten saber la información que se ha almacenado de acuerdo a los modelos. Las consultas son el paso previo a trabajar con las vistas y las plantillas, ya que permiten entregar la información que se desea.

2.5.5.8- Los formularios

Los formularios, permiten el ingreso de datos para su procesamiento, ya sea para crear nuevos contenidos, para modificar el contenido que ya está registrado previamente. Para crear los formularios se usa por convención un archivo nuevo llamado `forms.py` que se encuentra en la carpeta de la aplicación, donde se encuentran los archivos: `models.py` y `views.py`.

Para crear los formularios, es necesario importar al principio del archivo, el `ModelForm` para usar los modelos ya declarados en la aplicación, como ejemplo se explica el código de la Figura 2.10, en la que se exporta el `ModelForm`, así como el nombre del modelo a usar, una vez hecho esto, se declara una clase en la que dentro de esta se coloca el nombre y los campos del modelo.

```
from django.forms import ModelForm

from profesores_asignatura.models import puntos_conceptoA

class nivel_academicoForm(forms.ModelForm):

   class Meta:

     model = puntos_conceptoA


fiels=('licenciatura','especialidad','candidato_maestro')
```

Figura 2.10- Código para crear un formulario usando los datos de un modelo ya declarado

Capítulo 3
Diseño de Sistema

En éste capítulo, se muestra la manera en que se diseñaron algunas tablas utilizadas, para almacenar la información de la evaluación de las personas que participan en algún concurso de oposición. Así mismo, se presenta el diseño el de interfaz utilizando Django.

3.1- Diseño y creación de tablas

En el diseño de las tablas, es necesario identificar las entidades (tablas), las cuales son como un objeto, por ejemplo; una persona, lugar o cosa. Así mismo las tablas contienen sus atributos (campos) que se interpretan como las propiedades o características de la entidad.

Dada la naturaleza de éste trabajo, es necesario la tabla Profesor, en la que almacenara la información correspondiente. Es decir, nombre, apellidos y demás datos suficientes para mantener un control interno. La tabla con esta información se muestra en la Tabla 3.1.

Tabla 3.1- Diseño de la entidad Profesor

Profesor	
PK	Id_profesor
	Nombre_Profesor
	Apellido_Paterno
	Apellido_Materno
	Email
	Contraseña

En la Tabla 3.1 se crea, y se considera el campo Id_profesor, como una llave primaria, como una medida de referencia a las tuplas que forman la información ahí guardada.

En la entidad, Conceptos a evaluar, se almacenan los valores correspondientes a ello. Los conceptos que se evalúan, son diferentes dado que dependen del grado escolar obtenido por el profesor candidato. La tabla con esta información se muestra en la Tabla 3.2.

Tabla 3.2- Diseño de la entidad Conceptos a evaluar

Conceptos a evaluar	
PK	Id_Concepto
	Licenciatura
	Especialidad
	Candidato_maestro
	Maestría
	Candidato_Doctor
	Doctorado
	Profesor

En la Tabla 3.2 se crea, y se considera el campo Id_Concepto, como una llave primaria, como una medida de referencia a las tuplas que forman la información ahí guardada.

Teniendo en cuenta que al seleccionar alguna opción de la entidad Conceptos a evaluar, se debe subir un documento que avale el aspecto seleccionado, se crear la entidad Documento, para almacenar el documento correspondiente a ello. La tabla que permite guardar dicho documento se muestra en la Tabla 3.3.

Tabla 3.3- Diseño de la entidad Documento

Documento	
PK	Id_pdf
	Nombre_pdf
	pdf
	Profesor

En la Tabla 3.3 se crea, y se considera el campo Id_pdf, como una llave primaria, como una medida de referencia a las tuplas que forman la información ahí guardada.

En el campo Nombre_pdf se coloca el nombre del aspecto a evaluar, por ejemplo: Del aspecto nivel académico se evalúan los estudios formales realizados de licenciatura y posgrado en el área que se contrata. El nombre del pdf que es asignado por el sistema automáticamente es NA_ELP. El campo Profesor sirve para identificar a que profesor corresponde el documento guardado.

En las tablas creadas anteriormente, se puede observar que cada una tiene su llave primaria, por lo que ya se encuentra en la 1FN, aun así, existe redundancia entre las tablas ya que hay campos repetidos en las tablas, por lo cual es necesario normalizar para evitar esta redundancia y poder relacionarlas de manera correcta.

En las Tablas 3.2 y 3.3, se tienen el campo Profesor en común con la Tabla 3.1, por lo que es posible relacionar estas tablas, utilizando la llave primaria de la entidad Profesor de la Tabla 3.1, como llave foránea en la entidad Conceptos a evaluar y la entidad Documento, como se muestra en la Tabla 3.4 y 3.5.

Tabla 3.4- Rediseño de la entidad Conceptos a evaluar

Conceptos a evaluar	
PK	Id_Concepto
	Licenciatura
	Especialidad
	Candidato_maestro
	Maestría
	Candidato_Doctor
	Doctorado
FK	Id_profesor

Tabla 3.5- Rediseño de la entidad Documento

Documento	
PK	Id_pdf
	Nombre_pdf
	pdf
FK	Id_profesor

Ahora con el campo Id_profesor de la Tabla 3.4 y 3.5, se puede saber a qué profesor corresponden los Conceptos evaluados y el Documento, con esto se obtiene la dependencia funcional, por lo que las Tablas de la figura 3.4 y 3.5, se encuentran en la 2FN. De tal manera que no hay ningún atributo no primario de la tabla que dependa transitivamente de una llave primaria, por lo que así toma la 3FN.

3.2- Pantalla principal

El sistema contara con una pantalla de inicio como la que se muestra en la Figura 3.1**,** en la que se observan varias secciones, por lo que cada una cumple una función diferente; la sección de CONVOCATORIAS, se compone de un carousels, el cual es una presentación de diapositivas de contenido, la cual visualizara imágenes, las cuales en la parte inferior mostraran las convocatorias, así como una liga la cual mostrara toda la información referente a está.

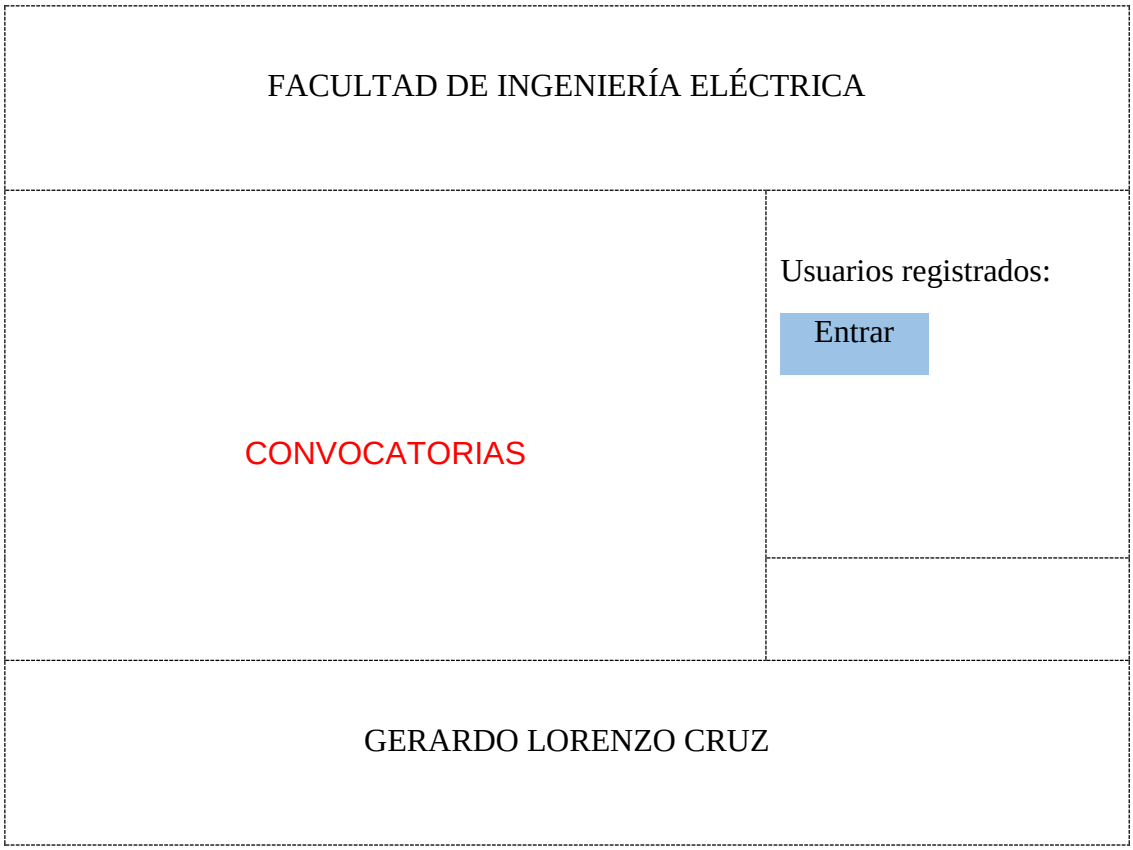


Figura 3.1- Pantalla principal del sistema

3.2.1- Botón Entrar

En la parte derecha se tendrá una sección llamada usuarios registrados, la cual tendrá un botón llamado Entrar, el botón permitirá acceder a cualquiera de los 3 tipos de usuarios, como lo es un profesor que se encuentre registrado por parte del administrador, un integrante de la comisión académica o el administrador; Para crear el botón **Entrar** que se observa en la Figura 3.1, será necesario colocar el código en el archivo index.html, que permita crear el botón y agregar la url a la cual se desee ir cuando se le dé clic, una vez creado el botón es necesario crear el login, el cual se verá como la Figura 3.2.

En el caso que acceda un profesor, podrá ingresar su información académica, es por ello que se visualizan las pantallas que deberán aparecer al ingresar al sistema, por lo que es necesario colocar el nombre de profesor y contraseña correspondiente. Teniendo en cuenta que se realiza de la misma manera para acceder como un integrante de la comisión académica o administrador, solo enviara a las pantallas de acuerdo al usuario.

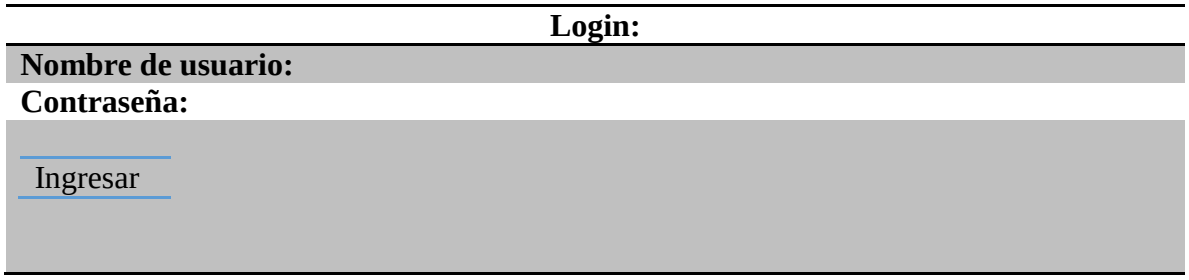


Figura 3.2- Acceso al sistema

En el caso de que el profesor ya se encuentra registrado por parte del administrador, este accederá al sistema, de lo contrario se redirección nuevamente a la parte de login de la Figura 3.2, por lo que es necesario acudir con el administrador para el previo registro; si ya se tuvo acceso, se visualizara una pantalla como la Figura 3.3, en la cual se tendrán los aspectos que son las características o cosas que se toman en cuenta a evaluar, como lo es nivel académico, experiencia académica, experiencia profesional, publicaciones, labor académica, resultados de exámenes; En la parte superior derecha se verá el nombre del profesor que ingreso al sistema.

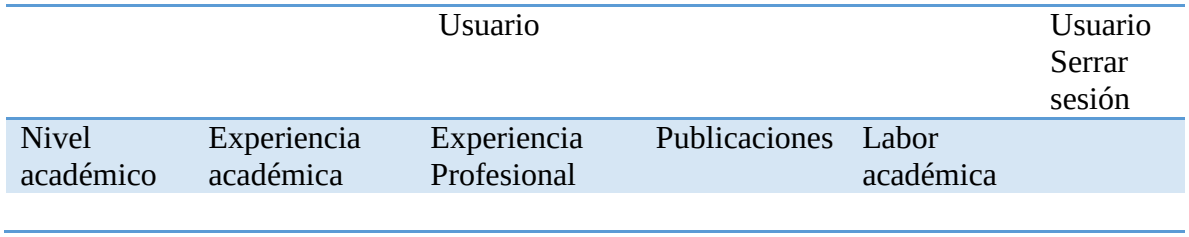


Figura 3.3- Pantalla con los aspectos a evaluar

Una vez visualizada la pantalla de la Figura 3.3, es necesario registrar la información, por lo que se realizará solo un ejemplo del procedimiento para capturar la información de las características de “Nivel académico” que se menciona en la Figura 3.3, ya que es el mismo procedimiento de captura de información para las características de experiencia académica, experiencia profesional, publicaciones y labor académica.

3.2.2- Nivel académico

Nivel académico despliega las opciones del menú de la Figura 3.4, que indican los aspectos que serán evaluados para el nivel académico: Estudios de licenciatura y posgrado en el área que se contrata, Cursos, talleres o seminarios de actualización profesional y Cursos, talleres o seminarios de actualización pedagógica.

		Usuario		Usuario Serrar sesión
Nivel académico	Experiencia académica	Experiencia Profesional	Publicaciones	Labor académica
Estudios de licenciatura y posgrado en el área que se contrata.				
Cursos, Talleres o Seminarios de actualización profesional				
Cursos, Talleres o Seminarios de actualización pedagógica				

Figura 3.4- Menú del nivel académico

3.2.2.1- Estudios de licenciatura y posgrado en el área que se contrata

La opción Estudios de licenciatura y posgrado en el área que se contrata del menú Nivel académico de la Figura 3.4, visualizara una pantalla como la que se muestra en la Figura 3.5, la cual se conforma de 3 partes: aspecto a considerar, documentos probatorios en el cual se subirá el documento que avala dicho aspecto considerado, este debe ser de acuerdo al punto por concepto seleccionado y la última parte se puede ver que en puntos por concepto se encuentran los niveles de estudio seguido de una O, está significa que son campos de selección pero esta es de acuerdo al documento que se subió.

Nivel académico		
Aspectos a considerar	Documentos Probatorios	Puntos por concepto
Estudios formales realizados de licenciatura y posgrado en el área que se contrata.	Documentos probatorios (en el área que se contrata) Subir Documento	Licenciatura: O Especialidad: O Candidato maestro: O Maestría: O Candidato Doctor: O Doctorado: O Guardar

Figura 3.5- Pantalla de captura de información de estudios formales realizados en licenciatura y posgrado en el área que se contrata

Para la parte de documentos probatorios, será necesario poder subir el documento que avale dicho aspecto a evaluar.

Tomando en cuenta que es el mismo procedimiento para subir el documento, solo se menciona en este apartado, ya que se realiza de la misma manera para todos los aspectos que son evaluados.

Se visualiza un botón llamado subir Documento como se muestra en la Figura 3.5, que al seleccionarlo permita ver una pantalla como la Figura 3.6, que tendrá un botón llamado examinar, que al seleccionarlo, abrirá el explorador de archivos para seleccionar el documento correspondiente a los puntos por concepto que se van a seleccionar, así como un botón submit que al seleccionarlo, guardara el documento previamente seleccionado en la carpeta documents, que se encuentra dentro de la carpeta media ubicada en la carpeta del proyecto tesis, el documento se guardara de la siguiente manera: carpeta del año, dentro de esta la carpeta del mes, dentro de esta la fecha y por último el documento.

En caso de no a ver seleccionado el documento y seleccionar el botón submit, aparecerá un mensaje el cual indica que es necesario selecciona un archivo.

Así para la parte de puntos por concepto que se observan en la Figura 3.5, serán una serie de opciones que correspondan al aspecto evaluado, en la que el profesor pueda seleccionar la opción de acuerdo al documento que se subió previamente.

Figura 3.6- Pantalla para subir documento

3.2.2.2- Cursos, talleres o seminarios de actualización profesional

La opción Cursos, Talleres o Seminario de actualización profesional del menú Nivel académico de la Figura 3.4, permite visualizar una pantalla como la que se muestra en la Figura 3.7, que permite registrar la información correspondiente.

Nivel académico		
Aspectos a considerar	Documentos probatorios	Puntos por concepto
Cursos, Talleres o Seminarios de actualización profesional.	Documentos probatorios Subir Documento	Evento fin: O Evento no afín: O Guardar

Figura 3.7- Pantalla de captura de información para cursos, talleres o seminarios de actualización profesional

Para los documentos probatorios y puntos por concepto se hacen de la misma manera que se mencionó anteriormente.

3.2.2.3- Cursos, talleres o seminario de actualización pedagógica

La opción Cursos, Talleres o Seminario de actualización pedagógica del menú Nivel académico de la Figura 3.4, permite visualizar una pantalla como la Figura 3.8, la que permite registrar la información correspondiente a evaluar.

Nivel académico		
Aspectos a considerar	Documentos probatorios	Puntos por concepto
Cursos, Talleres o Seminarios de actualización pedagógica.	Documentos probatorios Subir Documento	Evento fin: O Evento no afín: O Guardar

Figura 3.8- Pantalla de captura de información para cursos, talleres o seminario de actualización pedagógica

Para los documentos probatorios y puntos por concepto se hacen de la misma manera que se mencionó anteriormente.

Capítulo 4
Desarrollo y Pruebas del Sistema

En éste capítulo, se muestra la manera en que se creó el sistema. Primero se da énfasis a la visualización de la idea y, posteriormente, mostrando algunos códigos que fueron necesarios implementar para que la idea se llevara a cabo, se muestran sólo algunas partes, tanto de la visualización como del código, debido a que, casi el total del sistema se crea de manera semejante.

Así mismo, se presenta un ejemplo de prueba, haciendo un "recorrido" del sistema con los diferentes tipos de usuarios permitidos. Lo anterior, con la intención de que, éste capítulo, sea considerado como un "manual de usuario".

4.1- Usuarios

En la Figura 4.1, se muestra una imagen de cómo se ve la pantalla inicial del sistema. Se puede observar, en la parte inferior y centrada, una `sección` dedicada a las convocatorias de los concursos de oposición. En la parte derecha, en la `sección` de `usuarios registrados`, está el botón de "`Entrar`", el cual permite acceder al sistema. Se presentan los 3 tipos de usuarios a saber, "Profesor", "Administrador" o "Comisión académica".

Figura 4.1- Pantalla inicial

En la Figura 4.1, se observan 2 secciones. Una, donde se vea la imagen y la liga para acceder a los concursos de oposición, y dos, donde se vea el botón de usuarios registrados.

Para crear la primera `sección`, se hace uso de un "Carousel" el cual, utiliza código Bootstrap como el que se muestra en la Figura 4.2. Se hace uso de este tipo de código, debido a que permite crear el Carousel de forma simple. Este código, está contenido en el archivo index.html, el cual, está almacenado en la carpeta del proyecto.

```
<section class="main row"><!--row agrega fila-->
  <article class="col-sx-12 col-sm-8 col-md-9 col-md-9">
    <div id="carouselExampleIndicators" class="carousel slide" data-ride="carousel">
      <ol class="carousel-indicators">
        <li data-target="#carouselExampleIndicators" data-slide-to="0" class="active"></li>
        <li data-target="#carouselExampleIndicators" data-slide-to="1"></li>
        <li data-target="#carouselExampleIndicators" data-slide-to="2"></li>
      </ol>
      <div class="carousel-inner">
        <div class="carousel-item active">
          <img class="d-block w-100 " src="{% static 'imagen/fondo1.jpg' %}" alt="First slide">
          <div class="carousel-caption d-none d-md-block">
              <h5>Concurso de oposición</h5>
              <a href="{% static 'convocatorias_pdf/p1.pdf' %}">Convocatoria1.pdf</a>
            </div>
        </div>
        <div class="carousel-item">
          <img class="d-block w-100" src="{% static 'imagen/fondo2.jpeg' %}" alt="Second  slide">
          <div class="carousel-caption d-none d-md-block">
              <h5>Concurso de oposición</h5>
              <a href="{% static 'convocatorias_pdf/p2.pdf' %}">Convocatoria2.pdf</a>
          </div>
        </div>
        <div class="carousel-item">
          <img class="d-block w-100" src="{% static 'imagen/fondo3.jpeg' %}" alt="Third  slide">
          <div class="carousel-caption d-none d-md-block">
              <h5>Concurso de oposición</h5>
              <a href="{% static 'convocatorias_pdf/p3.pdf' %}">Convocatoria3.pdf</a>
          </div>
        </div>
      </div>
      <a class="carousel-control-prev" href="#carouselExampleIndicators" role="button" data-slide="prev">
        <span class="carousel-control-prev-icon" aria-hidden="true"></span>
        <span class="sr-only">Previous</span>
        <h5>Anterior</h5>
      </a>
```

Figura 4.2- Código para crear el Carousel

Para crear la segunda sección, se utiliza el código que se muestra en la Figura 4.3. Dentro de éste código, se observan varios elementos, llamados "etiquetas", que se utilizan para crear un código "html". Entre ellos están:

`<aside></aside>` el cual permite la inserción de una sección dentro de la ventana.

`<a></a>` permite crear un hiperenlace. Dentro de ésta etiqueta, se inserta el atributo `class="btn btn-primary btn-sm"`, para generar un botón pequeño de color azul.

El código mostrado en la Figura 4.3, está contenido en el archivo `index.html`.

```
<aside class="col-sx-12 col-sm-4 col-md-3 col-md-3  p-3 mb-2 bg-blue   text-dark"><!--p-3 mb-2 bg-warning -->
  <p  class="text-primary"> Usuarios registrados: </p>

    <a class="btn btn-primary btn-sm entrarg" href="{% url 'ingresar:usuario' %}" role="button">Entrar</a>

  <br>

</aside>
```

Figura 4.3- Generar sección usuarios registrados

4.1.1- Acceder como Profesor

En la parte derecha de la Figura 4.1, en la sección de `usuarios registrados` cuenta con el botón `Entrar`, que al seleccionarlo, visualiza la pantalla que se muestra en la Figura 4.4, por lo que es necesario colocar el usuario y contraseña para poder acceder al sistema como profesor con el nombre Rene, cabe mencionar que es el mismo procedimiento de login para cualquiera de los diferentes usuarios: Profesor, Administrador e Integrante de Comisión Académica.

Figura 4.4- Acceso al sistema

Para crear la pantalla de la Figura 4.4, se crea la aplicación `autentificación` en el proyecto, está aplicación contiene los archivos `urls.py y views.py`.

En el archivo `urls.py`, se coloca el código de la Figura 4.5, el cual indica que la `url` va interactuar con la función `ingresar_views` que valida que los usuarios estén registrados para acceder al sistema, la función se encuentra el archivo `views.py`.

```
from django.conf.urls import url, include
from autentificacion.views import ingresar_views,cerrar

urlpatterns = [
  url(r'^usuario$',ingresar_views,name="usuario"),
  url(r'^sesion$',cerrar,name="sesion")
]
```

Figura 4.5- url del login

El código agregado en el archivo `views.py` para validar que los usuarios estén registrados, es el de la Figura 4.6, que tiene la función `ingresar_views`, que se encarga de obtener los datos escritos en el formulario de la Figura 4.4, los valida y en el caso que sean correctos re direccionar el usuario a la pantalla correcta.

```
from __future__ import unicode_literals
from django.shortcuts import render
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
from django.contrib.auth import login, authenticate, logout
from django.contrib.auth.decorators import login_required
from django.shortcuts import render_to_response
from django.template import RequestContext
from django.http import HttpResponseRedirect
from django.contrib.auth.models import User

def ingresar_views(request):
    if request.method == 'POST':
        formulario = AuthenticationForm(request.POST)
        if formulario.is_valid:
            usuario = request.POST['username']
            clave = request.POST['password']
            print(usuario)
            print(clave)
            parametros=''
            acceso = authenticate(username=usuario , password=clave)
            if acceso is not None:
                if acceso.is_active:
                    login(request,acceso)
                    #redireccionar usuario valido
                    us=User.objects.get(username=usuario)
                    id=us.id
                    print("My IDD")
                    print(id)
                    parametros={
                        'usuario':usuario,
                        'id':id
                    }
                    return render(request,'profesor/pantalla2.html',parametros)
                    #return HttpResponseRedirect('autentificacion/noactivo.html')
                else:
                    #return render_to_response('autentificacion/noactivo.html', context_instance=RequestContext(request))
                    return render(request,'autentificacion/noactivo.html')
            else:
                #Validando administrador
                admin = "Gerardo"
                password = "admin_gerardo"
                parametros={
                    'admin':admin
                }
                if (usuario==admin) and (clave == password):
                    return render(request,'administrador/administrador.html',parametros)
                else:
                    #revision academica
                    revision1 = "Paco"
                    password1 = "academica_paco"
                    parametros={
                        'admin2':revision1
                    }
                    if (usuario==revision1) and (clave == password1):
                        return render(request,'revision_academica/revision_academica.html',parametros)

else:
    formulario = AuthenticationForm()
#return render_to_response('autentificacion/ingresar.html',{'formulario':formulario},'context_instance=RequestContext(requ
return render(request,'autentificacion/ingresar.html',{'formulario':formulario},'context_instance=RequestContext(request)'
```

Figura 4.6- Vista para validar usuarios registrados

Por ultimo se utiliza el codigo que se muestra en la Figura 4.7, para crear la pantalla de login de la Figura 4.4, para el acceso de los usuarios al sitema.

```
{% extends 'base/base.html' %}
{% block header%}
{% endblock%}
{% block content %}
<p align="center"><font size="10">LOGIN </font></p><!---->
<div class="container    "><!--p-3 mb-2 bg-info text-white-->
  <form id="formulario" method="post" action="">
  {% csrf_token %}
    <table align="center" >
    {{formulario}}
    </table>
    <br>
    <div class="centered">
      <button class="btn btn-primary right" type="submit">Ingresar</button>
      <!--<button type="submit">Guardar</button>-->
    </div>

  </form>
</div>
<br>
<style type="text/css">
  .centered{
  text-align: center;
  }

 </style>

{% endblock %}

{% block footer %}
{% endblock%}
```

Figura 4.7- Código para generar la pantalla de acceso al sistema

Ya que se ingresa el usuario y contraseña en el formulario de la Figura 4.4, y los datos son correctos, se visualiza la pantalla de la Figura 4.8, en la que se muestra el nombre del profesor que ingresó al sistema, los aspectos (Nivel académico, Experiencia académica, Experiencia profesional, Publicaciones, Labor académica) que se son considerados para la evaluación.

Figura 4.8- Pantalla de profesores

Para crear la pantalla de la Figura 4.8, en la carpeta templates ubicada en el proyecto tesis, se crea una carpeta llamada `profesor`, dentro de la carpeta `profesor` se crea un archivo llamado `pantalla2.html`, el archivo se compone de código Botstrap mostrado en la Figura 4.9, el que permite crear una barra de navegación más atractiva para el usuario.

```
<p align="center"> <font size="10">USUARIO </font></p>
<div class="container" >
<!--<p  id="head" align="center"><font size="5">Nombre:  {{usuario}}</font></p>-->
<nav id="navbar-example2" class="navbar navbar-expand-lg navbar-light bg-light">
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav mr-auto">
```

Figura 4.9- Código para crear la barra de navegación del profesor

4.1.1.1- Registro de información académica

Se realiza un ejemplo con `Estudios de Licenciatura y Posgrado` que corresponde al menú de `Nivel académico`, de la Figura 4.10, del procedimiento para el registro de información que corresponda a los aspectos que se muestran en la barra de navegación, ya que es el mismo para cualquier aspecto a evaluar.

Nivel académico despliega las opciones del menú, que indican los aspectos que son evaluados, como es, Estudios de licenciatura y posgrado, Cursos, Talleres o seminarios de actualización profesional y Cursos, Talleres o seminarios de actualización pedagógica, como se muestra en la Figura 4.10.

Figura 4.10- Menú de nivel académico

Para crear las opciones que despliega nivel académico, de la Figura 4.10, se utiliza el código de la Figura 4.11, que es una lista de enlaces para hacer referencia a los aspectos que se evalúan de nivel académico, el código se agrega dentro de la etiqueta `nav,` de la Figura 4.9.

```
<li class="nav-item dropdown">
<a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-haspopup="true"
    Nivel académico
  </a>
  <div class="dropdown-menu" aria-labelledby="navbarDropdown">
  <a class="dropdown-item" href="{% url 'crear:nuevo' %}?pk={{ id }}">Estudios de Licenciatura y Posgrado</a>
  <a class="dropdown-item" href="{% url 'crearb:b' %}?pk={{ id }}">Cursos,Talleres o Seminarios de actualización profesiona
  <a class="dropdown-item" href="{% url 'crearc:c' %}?pk={{ id }}" >Cursos,Talleres o Seminarios de actualizacion pedagógic
  </div>
</li>
```

Figura 4.11- Código para crear el menú de nivel académico

4.1.1.2- Estudios de Licenciatura y Posgrado

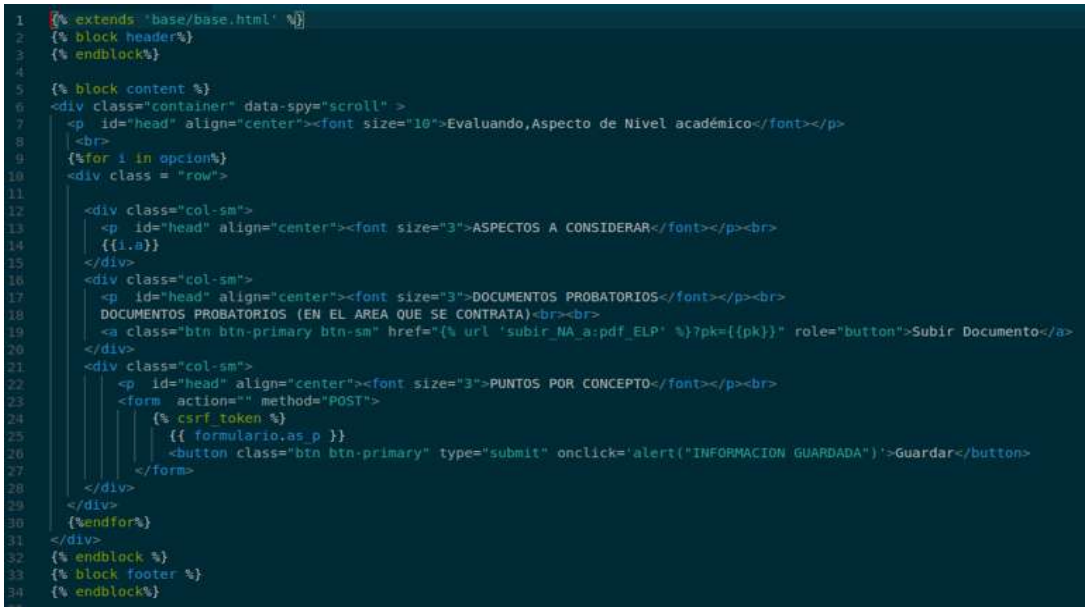
Estudios de Licenciatura y Posgrado del menú Nivel Académico mostrado en la Figura 4.10, visualiza una pantalla como la Figura 4.12, que permite ingresar la información que corresponde al aspecto que se evalúa.

Figura 4.12- Pantalla de captura de la información correspondiente a estudios de licenciatura y posgrado

En la Figura 4.12, es necesario almacenar el documento que avale el aspecto a considerar con respecto a los estudios de licenciatura y posgrado, para obtener un resultado satisfactorio.

Se explica el procedimiento para almacenar el documento de estudios formales realizados de licenciatura y posgrado en el área que se contrata, ya que es el mismo procedimiento para subir el documento para cualquier otro aspecto.

Para crear la pantalla de la Figura 4.12, se utiliza el código de la Figura 4.13, que muestra código HTML que se compone de etiquetas , que sirven para dividir la pantalla en 3 divisiones para colocar el aspecto que se evalúa, documento probatorio y los puntos por concepto.



```
{% extends 'base/base.html' %}
{% block header%}
{% endblock%}

{% block content %}
<div class="container" data-spy="scroll" >
  <p  id="head" align="center"><font size="10">Evaluando,Aspecto de Nivel académico</font></p>
  <br>
  {%for i in opcion%}
  <div class = "row">

    <div class="col-sm">
      <p  id="head" align="center"><font size="3">ASPECTOS A CONSIDERAR</font></p><br>
      {{i.a}}
    </div>
    <div class="col-sm">
      <p  id="head" align="center"><font size="3">DOCUMENTOS PROBATORIOS</font></p><br>
      DOCUMENTOS PROBATORIOS (EN EL AREA QUE SE CONTRATA)<br><br>
      <a class="btn btn-primary btn-sm" href="{% url 'subir_NA_a:pdf_ELP' %}?pk={{pk}}" role="button">Subir Documento</a>
    </div>
    <div class="col-sm">
        <p  id="head" align="center"><font size="3">PUNTOS POR CONCEPTO</font></p><br>
        <form  action="" method="POST">
          {% csrf_token %}
            {{ formulario.as_p }}
            <button class="btn btn-primary" type="submit" onclick='alert("INFORMACION GUARDADA")'>Guardar</button>
          </form>
    </div>
  </div>
  {%endfor%}
</div>
{% endblock %}
{% block footer %}
{% endblock%}
```

Figura 4.13- Código para crear la pantalla de nivel académico

A continuación, se explica el procedimiento para mostrar la Figura 4.12; Se agrega la `url`, en el apartado `urlpatterns` del archivo `urls.py` de la aplicación `profesores_asignatura`, como se muestra en la Figura 4.14.

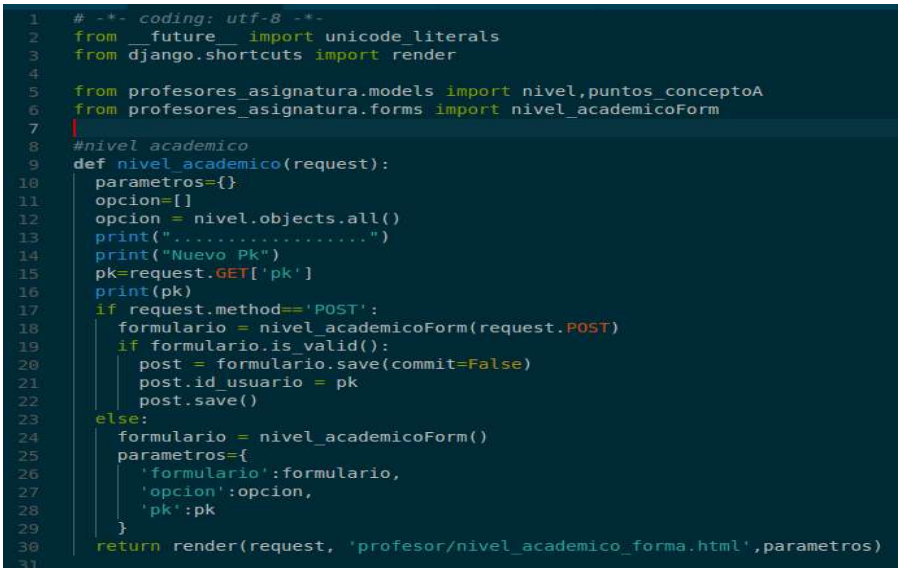
```
from django.conf.urls import url, include
from profesores_asignatura.views import nivel_academico,nivel_academicob,nivel_academicoC#,nivel_academicoList

from profesores_asignatura.views import experiencia_academicaA,experiencia_academicaB,experiencia_academicaC
from profesores_asignatura.views import experiencia_profesionalA,experiencia_profesionalB,experiencia_profesionalC,experienc
from profesores_asignatura.views import publicacionesa1, publicacionesa2, publicacionesa3, publicacionesa4, publicacionesa5,
from profesores_asignatura.views import labor_academicaA,labor_academicaB,labor_academicaC,labor_academicaD,labor_academicaE
from profesores_asignatura.views import resultados_examenesA,resultados_examenesB,resultados_examenesC
from profesores_asignatura.views import listar_upload,upload_file_NA_a,upload_file_NA_b,upload_file_NA_c
from profesores_asignatura.views import upload_file_EA_a,upload_file_EA_b,upload_file_EA_c
from profesores_asignatura.views import upload_file_EP_a, upload_file_EP_b, upload_file_EP_c, upload_file_EP_d
from profesores_asignatura.views import upload_file_P_a1,upload_file_P_a2,upload_file_P_a3,upload_file_P_a4,upload_file_P_a5
from profesores_asignatura.views import upload_file_LA_a, upload_file_LA_b, upload_file_LA_c, upload_file_LA_d, upload_file_

urlpatterns = [
  url(r'^nuevo$',nivel_academico,name='nuevo'),
  url(r'^b$',nivel_academicob,name='b'),
  url(r'^c$',nivel_academicoC,name='c'),
```

Figura 4.14- Código de la url

En el apartado de `urlpatterns` de la Figura 4.14, se agrega la `url(r'nuevo',nivel_academico,name='nuevo')`, que indica cuando se vaya a la `url` con el nombre `nuevo`, busca la función `nivel_académico` de la Figura 4.15, en el archivo `views.py` de la aplicación `profesores_asignatura`.



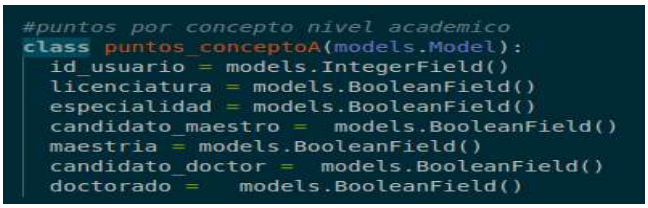
```python
# -*- coding: utf-8 -*-
from __future__ import unicode_literals
from django.shortcuts import render

from profesores_asignatura.models import nivel,puntos_conceptoA
from profesores_asignatura.forms import nivel_academicoForm

#nivel academico
def nivel_academico(request):
    parametros={}
    opcion=[]
    opcion = nivel.objects.all()
    print("..................")
    print("Nuevo Pk")
    pk=request.GET['pk']
    print(pk)
    if request.method=='POST':
        formulario = nivel_academicoForm(request.POST)
        if formulario.is_valid():
            post = formulario.save(commit=False)
            post.id_usuario = pk
            post.save()
    else:
        formulario = nivel_academicoForm()
        parametros={
            'formulario':formulario,
            'opcion':opcion,
            'pk':pk
        }
    return render(request, 'profesor/nivel_academico_forma.html',parametros)
```

Figura 4.15- Código de la función nivel_académico

En el código de la Figura 4.15, se exportar `render` para re direccionar a la plantilla `nivel_academico_forma.html` que se mencionó en la Figura 4.13, en la ruta `templates/profesor/nivel_academico_forma.html`, a continuación, se importa el modelo `puntos_conceptoA` de la Figura 4.16, del archivo models.py de la aplicación profesores_asignatura, el cual tiene los campos de los aspectos que se consideran para la evaluación.



```python
#puntos por concepto nivel academico
class puntos_conceptoA(models.Model):
    id_usuario = models.IntegerField()
    licenciatura = models.BooleanField()
    especialidad = models.BooleanField()
    candidato_maestro =  models.BooleanField()
    maestria = models.BooleanField()
    candidato_doctor =  models.BooleanField()
    doctorado =   models.BooleanField()
```

Figura 4.16- Modelo de puntos por concepto

Es necesario importar `ModelForm`, `forms`, así como el modelo de la Figura 4.16 para crear el formulario; primero se crea la clase `nivel_academicoForm` de la Figura 4.17, dentro de esta se crea la clase meta para definir en variables el modelo a usar y los campos que se visualizaran en el formulario para el ingreso de los datos del modelo de la Figura 4.16; la línea `exclude=('id_usuario')`, quiere decir que el campo `id_usuario` no se visualiza en el formulario.

Figura 4.17- Clase para generar el formulario

Dentro de la función `nivel_academico` de la Figura 4.15, se valida el formulario y si es válido lo manda mediante un diccionario de Python a la plantilla HTML de la Figura 4.13, la cual procesa y muestra los datos en el navegador.

4.1.1.3- Procedimiento para subir documento

Para crear la función del botón Subir Documento que se muestra en la Figura 4.12, que permite seleccionar y registrar el documento con el cual se comprueba el aspecto que se está evaluando, se crea el modelo `documentos_nivel_a` de la Figura 4.18, en el archivo `models.py` de la aplicación `profesores_asignatura`. El código significa que almacenara el identificador del profesor al que pertenece, el nombre que se le asigna automáticamente de acuerdo al aspecto que se esté evaluando y por último en el campo `pdf` almacena el documento seleccionado.

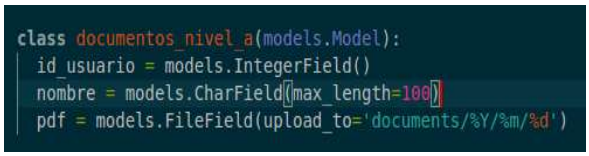


Figura 4.18- Modelo para guardar la información del Documento

Ya creado el modelo se agrega la `url` de la Figura 4.19, en el apartado `urlpatterns`, del archivo `urls.py,` de la aplicación profesores_asignatura.

El código de la Figura 4.19, significa que cuando se seleccione el botón Subir Documento, busca en el archivo `views.py` la función upload_file_NA_a de la Figura 4.20.

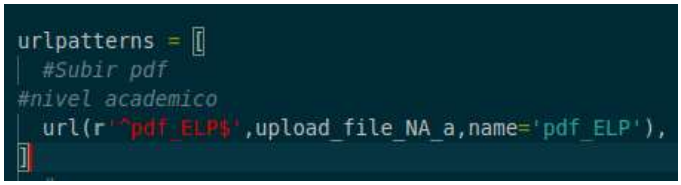


Figura 4.19- ulr para pdf

La Figura 4.20, muestra el codigo de la funcion upload_file_NA_a, que valida el documento en caso de averse selecionado o en caso de no seleccionar alguno, indica que es necesario seleccionar un documento.

```
from profesores_asignatura.models import documentos_nivel_a
def upload_file_NA_a(request):
    if request.method == 'POST':
        form = documentos_nivel_aForm(request.POST, request.FILES)
        print("..................")
        print("Nuevo Pk pdf")
        pk=request.GET['pk']
        print(pk)
        n="NA_ELP"
        if form.is_valid():
            instance=documentos_nivel_a(id_usuario = pk,nombre =n ,pdf = request.FILES['pdf'])#
            instance.save()
    else:
        form = documentos_nivel_aForm
    return render(request, 'profesor/upload.html', {'form': form})
```

Figura 4.20- Función para guardar Documento

Se crear en el archivo `forms.py` de la aplicación `profesores_asignatura`, una clase llamada `documentos_nivel_aForm`, como la Figura 4.21, que indica que se usa el modelo `documentos_nivel_a`, y se muestra el campo llamado `pdf`, se ocultaran los campos nombre, porque es asignado de manera automática y `id_usuario` al que pertenece.

```
class documentos_nivel_aForm(forms.ModelForm):
    class Meta:
        model = documentos_nivel_a
        fields = ('pdf',)
        exclude = ('nombre','id_usuario')
```

Figura 4.21- Clase para crear el formulario

Ya validados los datos en la función de la Figura 4.20, se mandan a la plantilla HTML, que son recibidos como se muestra en la Figura 4.22, observando que solo se crea el formulario con la etiqueta `{{form.as_p}}`.

```
{% extends 'base/base.html' %}
{% block header%}
<p  id="head" align="center"><font size="5">Subir Documento Probatorio</font></p>
{% endblock%}

{% block content %}
<br>
<div class="container">

<body>
    <form action="" method="post" enctype="multipart/form-data">
        {% csrf_token %}
        {{ form.as_p }}
        <input type="submit" value="Submit"><!-- onclick='alert("DOCUMENTO GUARDADO")'-->
    </form>
</div>
{% endblock %}

{% block footer %}
{% endblock%}
```

Figura 4.22- Plantilla HTML para mostrar los datos en navegador

El código mencionado anteriormente, permite visualizar la pantalla de la Figura 4.23, en la que se visualiza el botón examinar.

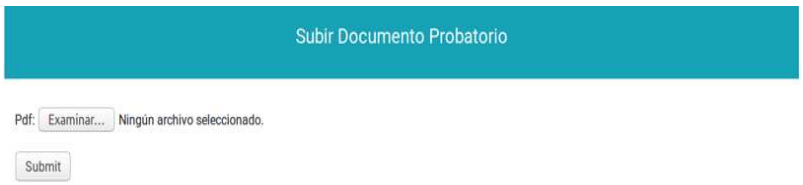


Figura 4.23- Subir Documento

El botón `Examinar` de la Figura 4.23, permite explorar los archivos y seleccionar el documento de interés para el profesor; una vez seleccionado el documento, es necesario tener a la vista una operación para guardarlo, esta se realiza dando clic en el botón `Submit`.

En caso de no haber seleccionado el documento y se selecciona el botón `Submit`, se visualiza una pantalla como la Figura 4.24, que indica que no se seleccionó ningún documento, por lo que es necesario seleccionar uno.

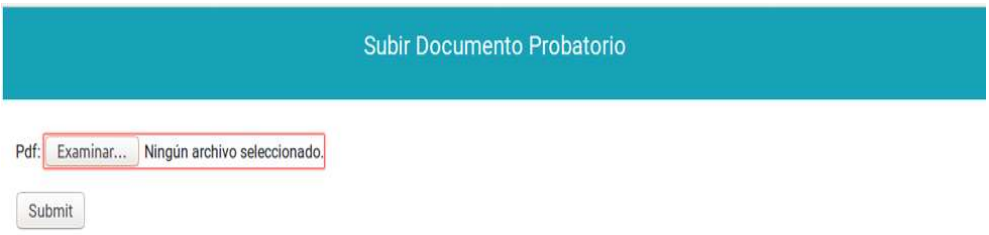


Figura 4.24- Documento no seleccionado

4.1.1.4- Procedimiento para crear el apartado de los puntos por concepto de la Figura 4.12. Se crea el modelo `puntos_conceptoA` de la Figura 4.25, dentro del archivo `models.py` de la aplicación `profesores_asignatura`, en el modelo se almacenan los aspectos que se evalúan, estos se presentaran en la pantalla de tal forma que el profesor pueda seleccionar el deseado.

```
#puntos por concepto nivel academico
class puntos_conceptoA(models.Model):
  id_usuario = models.IntegerField()
  licenciatura = models.BooleanField()
  especialidad = models.BooleanField()
  candidato_maestro =  models.BooleanField()
  maestria = models.BooleanField()
  candidato_doctor =  models.BooleanField()
  doctorado =   models.BooleanField()
```

Figura 4.25- Modelo de puntos a evaluar

Una vez creado el modelo mencionado en la Figura 4.25, en la aplicación `profesores_asignatura` se encuentra el archivo `urls.py`, en él se agrega en el apartado de `urlpatterns` la `url` de la Figura 4.26.

En la Figura 4.26, se importar la función `nivel_académico` que se encuentra en el archivo `views.py` de la misma aplicación, esto se hace para que se pueda utilizar dentro de `url(r'^nuevo',nivel_academico,name='nuevo'),` esto quiere decir que cuando se busque la `url`, se va a el archivo `views.py` de la misma aplicación y busca la función `nivel_academico` de la Figura 4.27.

```
from profesores_asignatura.views import nivel_academico
urlpatterns = [
  url(r'^nuevo$',nivel_academico,name='nuevo'),
]
```

Figura 4.26- url para mostrar los puntos por concepto que se están evaluando

En la Figura 4.27, se observa la función `nivel_académico`, en ella se realizan las operaciónes para validar si los datos del formulario son correctos, así como obtener el `pk` del profesor que ingreso al sistema, para mandarlo y posteriormente guardarlo en la tabla de `puntos_conceptoA` mostrada en la Figura 4.25, para saber a qué profesor corresponden los datos y por ultimo mandarlos a la plantilla HTML que se muestra en la Figura 4.29.

```
#nivel academico
def nivel_academico(request):
    parametros={}
    opcion=[]
    opcion = nivel.objects.all()
    pk=request.GET['pk']
    print(pk)
    if request.method=='POST':
        formulario = nivel_academicoForm(request.POST)
        if formulario.is_valid():
            post = formulario.save(commit=False)
            post.id_usuario = pk
            post.save()
    else:
        formulario = nivel_academicoForm()
        parametros={
            'formulario':formulario,
            'opcion':opcion,
            'pk':pk
        }
    return render(request, 'profesor/nivel_academico_forma.html',parametros)
```

Figura 4.27- Función para validar los datos

Para mostrar el formulario que agrega la información al modelo de la Figura 4.25, él se valida en la función nivel_academico de la Figura 4.27, se agrega el código de la Figura 4.28, en la que se especifica el modelo a usar, en este caso es puntos_conceptoA de la Figura 4.25, los campos que se visualizan del modelo de la Figura 4.25 como lo es: licenciatura, especialidad, candidato a maestro, maestría, candidato a doctor, doctorado.

Por último, los campos que no se visualizaran que en este caso es el id_usuario, ya que este valor es asignado desde la función nivel_academico de la Figura 4.27.

```
from django.forms import ModelForm
from django import forms
from profesores_asignatura.models import nivel,puntos_conceptoA

class nivel_academicoForm(forms.ModelForm):
    class Meta:
        model = puntos_conceptoA
        fields = ('licenciatura', 'especialidad','candidato_maestro','maestria','candidato_doctor','doctorado')
        exclude = ('id_usuario',)
```

Figura 4.28- Código para crear el formulario

Por último la plantilla HTML de la Figura 4.29, se compone de etiquetas Django, que recibe los datos que son enviados de la función `nivel_academico` de la Figura 4.27. El archivo se compone de tres divisiones principalmente: permiten mostrar la característica del nivel académico que se esté evaluando, apartado donde aparece el botón para subir el documento que avale lo puntos por concepto que se seleccionen y por ultimo las opciones de puntos por concepto, cabe mencionar que por cada opción de puntos por concepto se debe subir un documento que lo avale.

Figura 4.29- Plantilla HTM

4.1.2- Administrador

El administrador, registra los Profesores asignándoles nombre de usuario y contraseña para que ingresen al sistema y registren su información académica.

El administrador también puede realizar las funciones:

- Editar los datos de los profesores.
- Eliminar profesores.
- Ver historial de los profesores.

4.1.2.1- Acceder como Administrador

La Figura 4.1, muestra la pantalla principal del sistema, al seleccionar el botón Entrar se muestra la Figura 4.4, en la que se coloca el usuario y contraseña que corresponda al administrador, en este caso se utiliza:

Usuario: Gerardo

Contraseña: admin_gerardo

Una vez que se validan los datos y son correctos, se muestra la Figura 4.30, en la que se observa dos botones: Crear Usuario, Ver Usuarios y en la parte superior derecha se observa el nombre Gerardo, al seleccionarlo se despliega una opción para cerrar sesión.

Figura 4.30- Administrador

Para crear la pantalla de la Figura 4.30, se utiliza el código de la Figura 4.31, que se compone de código Bootstrap y etiquetas Django, para darle mejor presentación a la pantalla. En el código se observa las url a las cuales se direcciona cuando se selecciona el botón crear usuario o ver usuario.

```
{% extends 'base/base.html' %}
{% load staticfiles %}
{% block header%}
<p align="center"> <font size="10">ADMINISTRADOR</font></p>
<nav id="navbar-example2" class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <div class="container-fluid">
      <table class="table table-striped">
        <tbody>
          <tr>
            <td>
              <a class="btn btn-primary" href="{% url 'usuario:registrar' %}"> Crear Usuario</a>
            </td>
            <td>
              <a class="btn btn-primary" href="{% url 'historial:user'%}"> Ver Usuarios</a>
            </td>
          </tr>
        </tbody>
      </table>
    </div>

    <li class="nav-item dropdown">
      <img class="d-block w-10 " width="50" height="50" src="{% static 'imagen/usuario.png' %}" alt="First slide">
      <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-haspopup="
        {{admin}}
      </a>
      <div class="dropdown-menu" aria-labelledby="navbarDropdown">
        <a class="dropdown-item" href="{% url 'cerrar:sesion' %}" > Cerrar sesión</a>
      </div>
    </li>
</div>
</nav>
{% endblock%}
```

Figura 4.31- Código para generar la pantalla del administrador

4.1.2.2- Botón Crear Usuarios (Profesor)

El botón Crear Usuario, permite registra la información de los profesores que deseen participar por alguna materia o plaza vacante, como es nombre de usuario y contraseña, para poder ingresar al sistema.

Para crear la función del botón, se agrega la url de la Figura 4.32, en el archivo urls.py de la aplicación usuarios, la función que hace la url cuando se selecciona el botón, busca la función Registro Usuario en el archivo views.py de la Figura 4.33.

```python
from django.conf.urls import url

#importar la vista
from usuarios.views import RegistroUsuario
urlpatterns = [
  url(r'^registrar$',RegistroUsuario.as_view(),name="registrar")
]
```

Figura 4.32- url para crear usuarios

La Figura 4.33, muestra el código utilizado para el registro de la información de los profesores, en él se importa el modelo `User`, que es la tabla donde se guarda el nombre, apellidos y email del profesor. La clase `registroForm` de la Figura 4.34, es un formulario que permite ingresar la información a la tabla `User`, la clase `registroForm` se importa del se importa del archivo `forms.py`.

```python
from __future__ import unicode_literals
from django.core.urlresolvers import reverse_lazy
from django.views.generic import CreateView
from django.contrib.auth.models import User
from usuarios.forms import registroForm

class RegistroUsuario(CreateView):
  model = User
  template_name = "usuarios/registrar.html"
  form_class = registroForm
  success_url = reverse_lazy('historial:user')
```

Figura 4.33- Función RegistroUsuario

En el código de la Figura 4.34 se importa el modelo User de models, se crea una clase llamada registroForm que dentro de esta se escriben los campos que se visualizan en el formulario para ingresar la información.

```python
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm

class registroForm(UserCreationForm):
  class Meta:
    model = User
    fields = [
      'username',
      'first_name',
      'last_name',
      'email',
    ]
    labels ={
      'username' : 'Nombre de Usuario',
      'first_name': 'Primer apellido',
      'last_name' : 'Segundo apellido',
      'email' : 'Correo',
    }
```

Figura 4.34- Clase registroForm

Para que el formulario pueda visualizarse se utiliza la platilla de la Figura 4.35, que se compone de etiqueta Django y código Bootstrap.

```
{% extends 'base/base.html' %}
{% block header%}
{% endblock%}
{% block content %}
<p  id="header" align="center"><font size="10">..... </font></p>
<div class="container">
  <form method="post">
    {% csrf_token %}
    <table align="center">
      {{ form.as_p }}
    </table>
    <button class="btn btn-primary" type="submit">Guardar</button>
  </form>

</div>
<br>
{% endblock %}
{% block footer %}
{% endblock%}
```

Figura 4.35- Plantilla HTML

Para verificar los usuarios que participan por alguna materia o plaza vacante, seleccionar el botón Ver Usuarios de la Figura 4.30, obteniendo los usuarios registrados en el sistema como se muestra en la Figura 4.36, por lo tanto, el profesor ya puede registrar la información.

Figura 4.36- Profesores registrados

Para crear la función del botón Ver Usuarios, se agrega en el archivo `urls.py` de la aplicación `administrador` la `url` de la Figura 4.37, que indica que cuando se seleccione el botón, busca en el archivo `views.py` de la aplicación `administrador`, la función `historial_usuario` la cual se ve en la Figura 4.38.

```
from django.conf.urls import url, include
from administrador.views import historial_usuario

urlpatterns = [
  url(r'^user$',historial_usuario,name="user"),

]
```

Figura 4.37- url para la función del botón Ver Usuarios

La Figura 4.38, muestra la función `historial_usuario`, que obtiene mediante una consulta los profesores registrados en la tabla `User`, y los manda a la plantilla `historial_usuarios.html`.

```python
from __future__ import unicode_literals
from django.shortcuts import render
from django.contrib.auth.models import User
from django.views.generic import ListView,CreateView,UpdateView,DeleteView
from usuarios.forms import registroForm
from django.core.urlresolvers import reverse_lazy

def historial_usuario(request):
    usuarios =  User.objects.all()
    print(usuarios)
    parametros = {
        'usuarios':usuarios
    }
    return render(request,'administrador/historial_usuarios.html',parametros)
```

Figura 4.38- Función historial_usuario

El código de la Figura 4.39, muestra cómo se reciben los datos enviados de la función historial_usuario que se muestra en la Figura 4.38.

```
{% extends 'base/base.html' %}
{% block header%}
{% endblock%}
{% block content %}
<p align="center"><font size="10">USUARIOS REGISTRADOS</font></p>
<br>
<div class="container" data-spy="scroll" >
  <table class="table table-striped">
    <thead>
      <tr>
        <td>Nombre</td> <td>Apellido paterno</td> <td>Apellido materno</td> <td>Correo</td>
        <td>Historial</td> <td>Editar</td> <td>Eliminar</td>
      </tr>
    </thead>
    <tbody>
      {% if usuarios %}
      {% for usuario in usuarios%}
        <tr>
          <td>{{usuario.username}}</td> <td>{{usuario.first_name}}</td><td>{{usuario.last_name}}</td><td>{{usuario.email}}
          <td><a class="btn btn-primary btn-sm" href="{% url 'pantallatres:poner' usuario.id %}">Historial</a></td>
          <td><a class="btn btn-primary btn-sm" href="{% url 'editar_usuario' usuario.id %}">Editar</a></td>
          <td><a class="btn btn-danger btn-sm" href="{% url 'eliminar_usuario' usuario.id %}">Eliminar</a></td>
        </tr>
      {% endfor%}
      {% else%}
      <h1> No hay usuarios registrados</h1>
      {%endif%}
    </tbody>
  </table>
</div> {% endblock %}{% block footer %}{% endblock%}
```

Figura 4.39- Plantilla para mostrar los profesores registrados

Para registrar el profesor (José), se selecciona el botón Crear Usuario de la Figura 4.30, obteniendo la pantalla de la Figura 4.40, en la que es necesario ingresar los datos del profesor: nombre de usuario, apellidos, correo y contraseña.

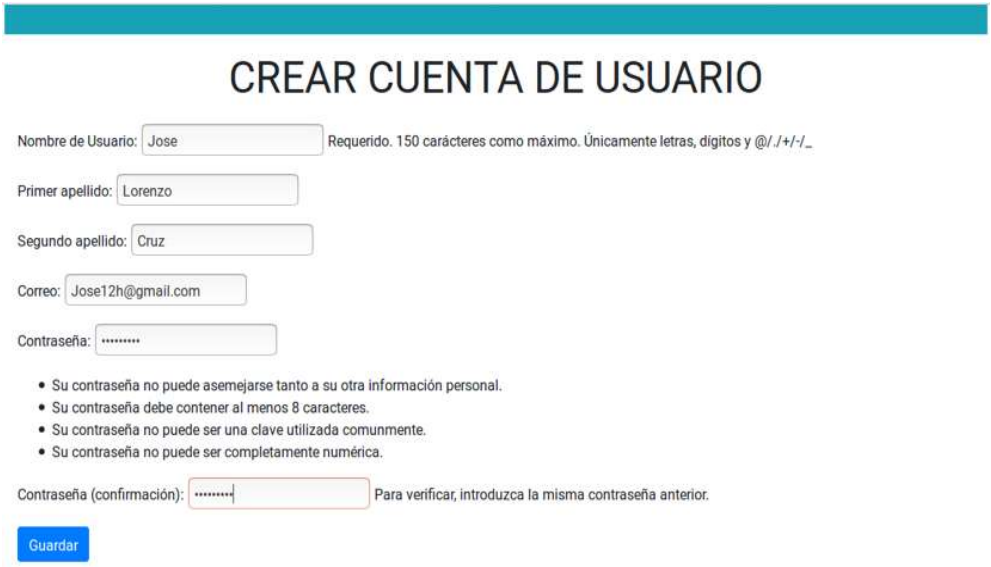


Figura 4.40- Crear cuenta de usuario (profesor)

Ya que se han llenado los campos, Nombre de Usuario, Apellidos, Correo y Contraseña mostrados en la Figura 4.40, seleccionar el botón Guardar y con esto se da por terminado el registro, por lo tanto, se visualizará la pantalla de la Figura 4.30.

Para verificar que el profesor se registró correctamente, seleccionar el botón Ver Usuarios de la Figura 4.30, visualizando los profesores ya registrados como se ve en la Figura 4.41, observando que ya se encuentra el nombre del profesor (José) el cual se registró.

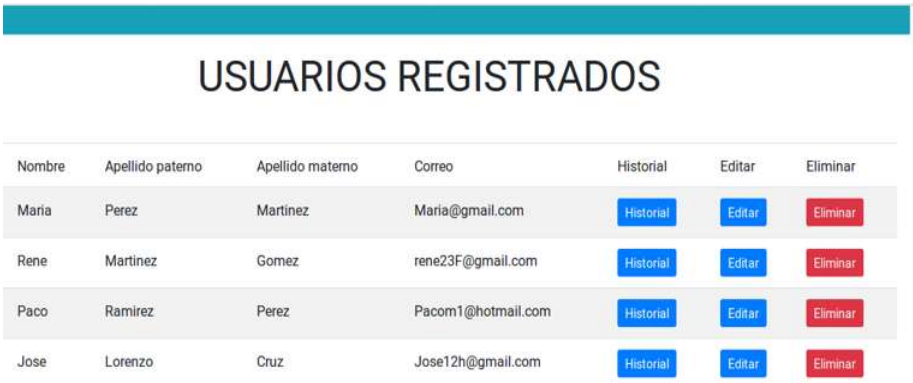


Figura 4.41- Profesores registrados

4.1.2.3- Botón Editar Usuario (Profesor)

Se seleccionar el botón Editar de la Figura 4.41, que corresponde al profesor que se desea modificar la información, en el caso que se ha registrado la información del profesor erróneamente o para actualizar la información.

Ejemplo: Editar los datos que corresponden a la profesora María Pérez Martínez.

Al seleccionar el botón Editar de la Figura 4.41, se observan los campos: nombre, apellidos, correo, y contraseña como se muestra en la Figura 4.42, ya que se ha modificado la información, es necesario seleccionar el botón Guardar para registrar loa datos.

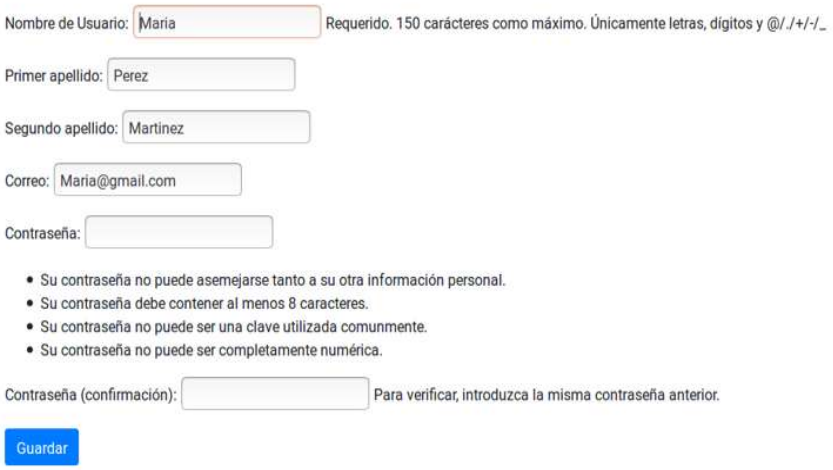


Figura 4.42- Editar profesor

4.1.2.4- Eliminar Usuario (Profesor)

Al seleccionar el botón Eliminar de la Figura 4.41, que corresponda al profesor que ya no participe por la materia o plaza vacante, se elimina el registro del profesor de la base de datos, por lo que el profesor ya no tendrá acceso al sistema.

Ejemplo: Para eliminar a la profesora María Pérez Martínez, se selecciona el botón Eliminar, por lo que se visualiza la pantalla de la Figura 4.43, que valida si se desea eliminar a María Pérez Martínez.

Figura 4.43- Eliminar profesor

Al seleccionar la opción (Si, Eliminar) de la Figura 4.43, se elimina María Pérez Martínez de la base de datos**,** como se observando en la Figura 4.44, ya no aparece, con esto ya no tiene acceso María Pérez Martínez al sistema hasta que se registre nuevamente.

Figura 4.44- Profesores registrados después de eliminar a María Pérez Martínez

4.1.2.5- Historial de Usuarios (Profesores)

Al seleccionar el botón Historial de José de la Figura 4.44, se visualiza la pantalla de la Figura 4.45, en la que se observa la información de nivel académico, experiencia académica, experiencia profesional, publicaciones, labor académica, resultados de exámenes de José.

Al seleccionar una de las opciones de la Figura 4.45, se despliega el menú de cada una para poder ver toda la información registrada por José.

Figura 4.45- Historial del profesor

4.1.3- Comisión Académica

La función de la comisión académica, es visualizar la información registrada por cada profesor, para poder asignar un puntaje de acuerdo a los puntos por concepto y documentos que registro el profesor.

4.1.3.1- Acceder como integrante de la Comisión Académica

En la Figura 4.1, se muestra la pantalla inicial del sistema, la cual permite ingresar como integrante de la comisión académica.

Al seleccionar el botón Entrar, se muestra la Figura 4.4, en la que se coloca el usuario y contraseña del integrante de la comisión académica, que en este caso es:

Usuario: Paco

Contraseña: academica_paco

Una vez que se validan los datos y se ingresa al sistema, se obtiene la Figura 4.46, en la que se observa un botón llamado VER USUARIOS E HISTORIAL, en la parte superior derecha, el nombre del integrante de la comisión académica, que al seleccionarlo permite cerrar la sesión, por lo que se visualiza la Figura 4.1.

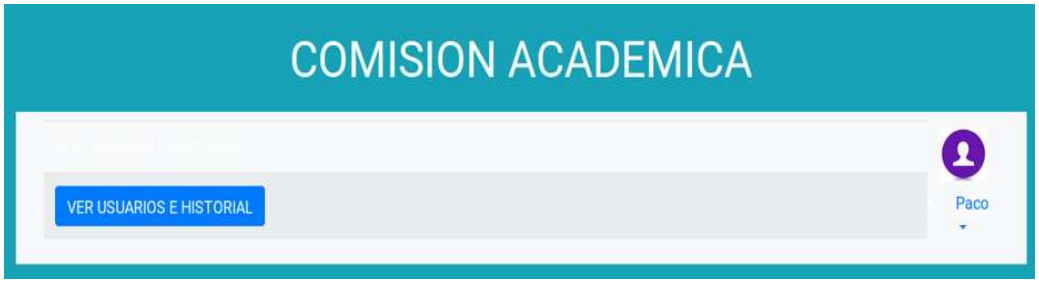


Figura 4.46- Comisión académica

4.1.3.2- Botón Ver Usuarios e Historial

Al seleccionar el botón ver usuarios e historial, se obtiene la Figura 4.47, en la que se visualizan los profesores registrados, así como un botón de Historial.

Figura 4.47- Profesores registrados

4.1.3.3- Botón Historial

Al seleccionar el botón Historial de René**,** se visualiza la Figura 4.48, observando en la parte superior, el nombre del usuario: René, de quien está viendo su historial, así como los aspectos que son evaluados: nivel académico, experiencia académica, experiencia profesional, publicaciones, labor académica.

La opción CALIFICAR se refiere a la asignación de una calificación a los aspectos que se están evaluando, por ejemplo, Examen oral sobre la materia, Desarrollo del tema de concurso, Exposición del tema frente al grupo, estos aspectos sirven para evaluar a el profesor que está participando por alguna materia o plaza vacante dentro de la Facultad de Ingeniería Eléctrica.

La opción resultados de exámenes del menú CALIFICAR, permiten ver la calificación de los profesores de exámenes realizados.

Figura 4.48- Historial de Rene

4.1.3.4- Nivel académico

Al seleccionar la opción Nivel académico del menú que se despliega, se obtiene la Figura 4.49, en la que se ven los aspectos a evaluar (estudios de licenciatura y posgrado, Cursos, talleres o seminarios de actualización profesional, Cursos, talleres o seminarios de actualización pedagógica).

Figura 4.49- Aspectos a evaluar

Tomando en cuenta que es el mismo procedimiento para cada aspecto a considerar (nivel académico, experiencia académica, experiencia profesional, publicaciones, labor académica).

Solo se presenta un ejemplo de cómo son las pantallas que se obtienen en el caso de que se tengan datos registrados, de igual manera en el caso de que no se han registrado datos.

Se observan las pantallas como se mencionan a continuación, solo variando los aspectos a considerar y los valores correspondientes.

4.1.3.4.1- Estudios de Licenciatura y Posgrado

Al seleccionar la opción Estudios de licenciatura y posgrado, se obtiene la Figura 4.50, en la que se observan que Rene tiene especialidad y sube el documento que avala la especialidad, el documento se guarda en la carpeta `media/documents/...`.

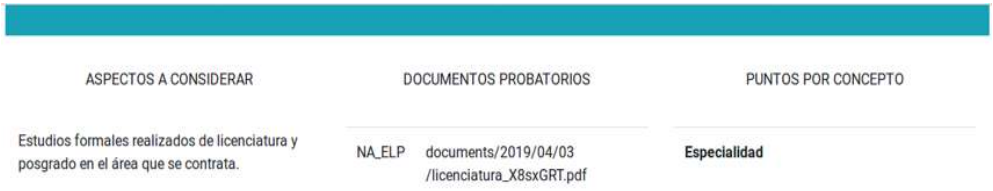


Figura 4.50- Estudios de licenciatura y posgrado

4.1.3.4.2- Cursos, Talleres o Seminarios de Actualización Profesional

Al seleccionar la opción Cursos, talleres o seminarios de actualización profesional, se obtiene la Figura 4.51, en la que se observa que no tiene datos registrados en este aspecto.

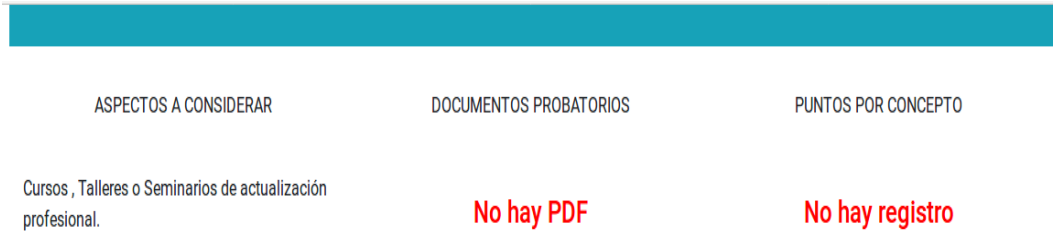


Figura 4.51- Cursos, talleres o seminario de actualización profesional

4.1.3.5- Calificar

La opción Calificar de la Figura 4.52, se compone de 3 aspectos a evaluar, el examen oral, desarrollo del tema, exposición frente al grupo, así como la opción de RESULTADOS DE EXAMENES que muestra la calificación de cada aspecto evaluado.

Figura 4.52- Calificar

4.1.3.5.1- Examen Oral sobre (Materia o Área Académica)

Al seleccionar Examen oral de la materia o área académica, se obtiene la Figura 4.53, en la que se observan los aspectos que se evalúan en el examen oral, así como los campos en donde se coloca el puntaje correspondiente a cada aspecto.

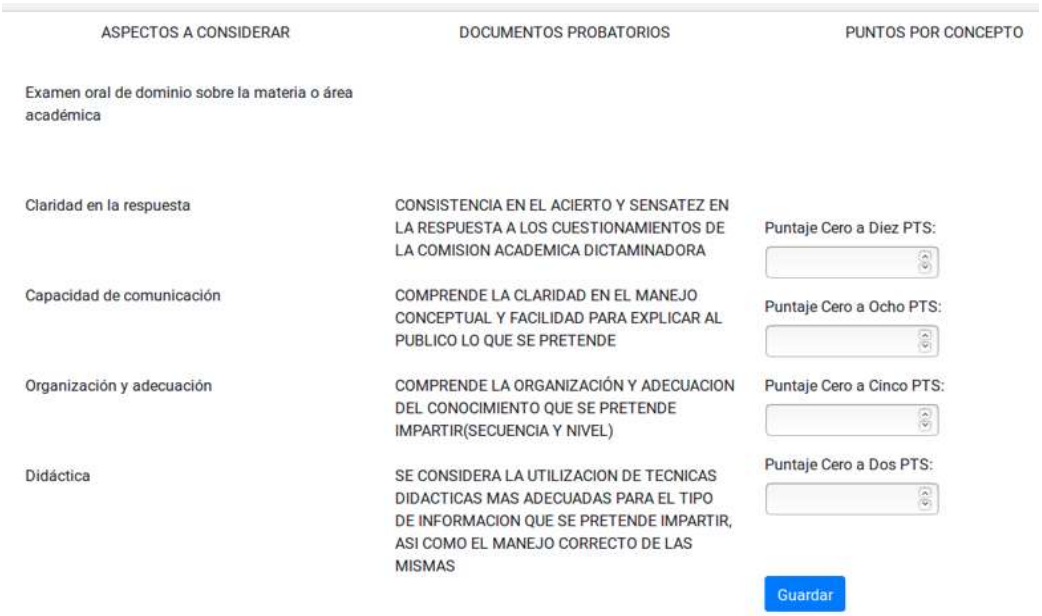


Figura 4.53- Calificar examen oral

4.1.3.5.2- Desarrollo del Tema del Concurso

Al seleccionar la opción Desarrollo del tema del concurso, se obtiene la Figura 4.54, en la que se observan los campos para coloca el puntaje correspondiente de acuerdo como presento el tema.

Figura 4.54- Calificar desarrollo del tema

4.1.3.5.3- Exposición del Tema Frente al Grupo

Al seleccionar la opción exposición del tema frente al grupo, se obtiene la Figura 4.55, en la que se observan los campos para seleccionar la calificación correspondiente de acuerdo como se expresó en la exposición del tema.

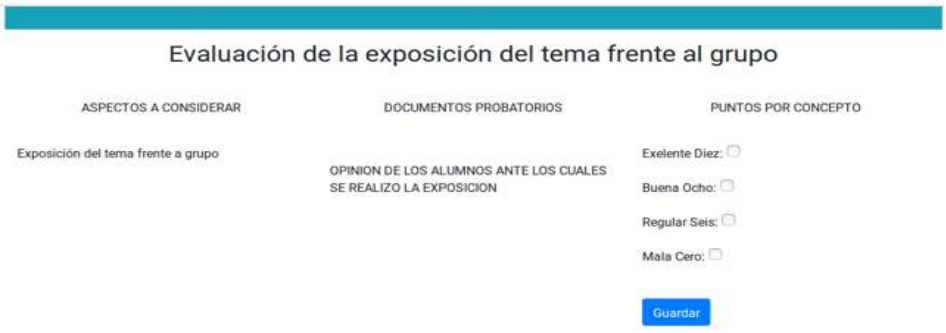
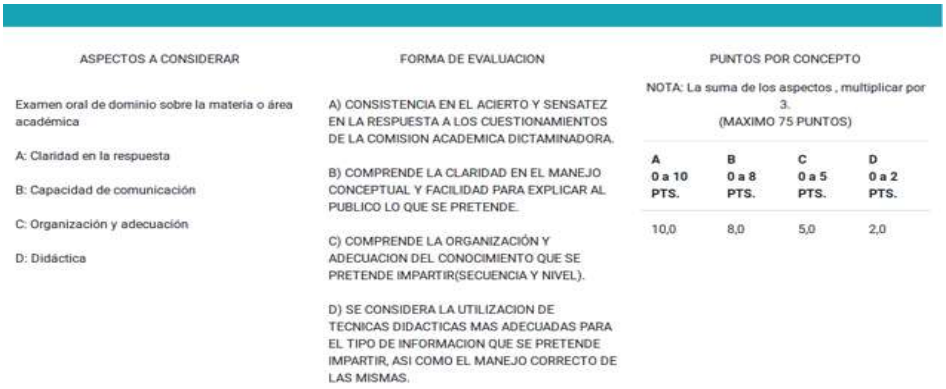


Figura 4.55- Calificar exposición

4.1.3.5.4- Resultados de Exámenes

La opción Resultados de exámenes de la Figura 4.52, muestra el puntaje obtenido por el profesor René, como se observa en las Figuras siguientes.

La Figura 4.56, muestra el puntaje obtenido por parte del profesor durante el examen oral en el que se evalúa la claridad de las respuestas, capacidad de comunicación, organización y la organización de técnicas didácticas más adecuadas para el tipo de información que se desea impartir.



ASPECTOS A CONSIDERAR	FORMA DE EVALUACION	PUNTOS POR CONCEPTO
Examen oral de dominio sobre la materia o área académica A: Claridad en la respuesta B: Capacidad de comunicación C: Organización y adecuación D: Didáctica	A) CONSISTENCIA EN EL ACIERTO Y SENSATEZ EN LA RESPUESTA A LOS CUESTIONAMIENTOS DE LA COMISION ACADEMICA DICTAMINADORA. B) COMPRENDE LA CLARIDAD EN EL MANEJO CONCEPTUAL Y FACILIDAD PARA EXPLICAR AL PUBLICO LO QUE SE PRETENDE. C) COMPRENDE LA ORGANIZACIÓN Y ADECUACION DEL CONOCIMIENTO QUE SE PRETENDE IMPARTIR(SECUENCIA Y NIVEL). D) SE CONSIDERA LA UTILIZACION DE TECNICAS DIDACTICAS MAS ADECUADAS PARA EL TIPO DE INFORMACION QUE SE PRETENDE IMPARTIR, ASI COMO EL MANEJO CORRECTO DE LAS MISMAS.	NOTA: La suma de los aspectos , multiplicar por 3. (MAXIMO 75 PUNTOS)

A 0 a 10 PTS.	B 0 a 8 PTS.	C 0 a 5 PTS.	D 0 a 2 PTS.
10,0	8,0	5,0	2,0

Figura 4.56- Resultado del examen oral

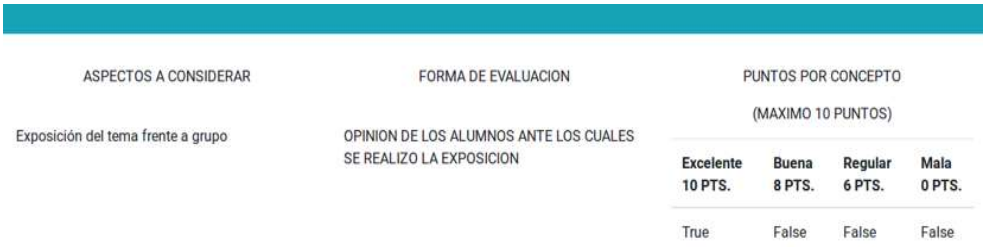
La Figura 4.57, muestra la calificacion que se obtuvo durante la evaluacion del desarrollo del tema, el cual se presenta por escrito.

ASPECTOS A CONSIDERAR	FORMA DE EVALUACION	PUNTOS POR CONCEPTO (MAXIMO 15 PUNTOS)
Desarrollo por escrito del tema objeto del concurso	PRESENTAR EL TRABAJO POR ESCRITO	

Portada 0 a 1.0 PTS.	Justificacion y relevancia 0 a 1.5 PTS.	Objetivo del tema 0 a 2.0 PTS.	Desarrollo del tema 0 a 8.0 PTS.	Conclusiones 0 a 1.5 PTS.	Bibliografia 0 a 1.0 PTS.
1,0	1,5	2,0	8,0	1,5	1,0

Figura 4.57- Resultado del tema desarrollado por escrito

La Figura 4.58, muestra los resultados de las exposiciones realizadas frente la grupo, por parte del profesor.



ASPECTOS A CONSIDERAR	FORMA DE EVALUACION	PUNTOS POR CONCEPTO (MAXIMO 10 PUNTOS)
Exposición del tema frente a grupo	OPINION DE LOS ALUMNOS ANTE LOS CUALES SE REALIZO LA EXPOSICION	

Excelente 10 PTS.	Buena 8 PTS.	Regular 6 PTS.	Mala 0 PTS.
True	False	False	False

Figura 4.58- Resultado de las exposiciones

Capítulo 5
Conclusiones y Trabajos Futuros

5.1- Conclusiones

Se cumplen los objetivos planteados al inicio del proyecto, ya que la interfaz permite el ingreso a cualquiera de los 3 tipos de usuario, almacenar los documentos de los profesores que participan en concurso, para que en cursos posteriores no sea necesario presentar toda la documentación nuevamente.
Finalmente, queda la satisfacción de haber trabajado en el diseño e implementación de un sistema web que aporta una ayuda en una de las múltiples tareas que realiza la administración de la facultad.

5.2- Ideas para el Sistema Web en un Futuro

Por el interés que hay por este tipo de aplicaciones, hay algunas tareas que aún se le pueden añadir al sistema que pueden servir como continuación a este proyecto. A continuación, se mencionan algunas:

1. Crear un módulo para generar reportes históricos de concursos para un profesor.
2. Implementar módulos de código para que se automatice la generación de las actas correspondientes, generales e individuales.

Bibliografía

[1] D. p. indexDesarrollo, «indexDesarrollo,» Copyright 2017. [En línea]. Available: Copyright 2017 | Todos los derechos resevados | Diseñado por indexDesarrollo.

[2] A. Fenollosa, «PROGRAMADOR WEB VALENCIA,» 2018. [En línea]. Available: https://programadorwebvalencia.com/cual-es-el-mejor-sistema-operativo-para-programar/.

[3] M. P. Web, «miposicionamientoweb.es,» [En línea]. Available: https://miposicionamientoweb.es/que-es-un-dominio/.

[4] E. S.L., «Doominio.com,» 2003-2018. [En línea]. Available: https://blog.doominio.com/hosting-definicion-significado/. [Último acceso: 2018].

[5] wikipedia, «wikipedia.org,» 21 Noviembre 2018. [En línea]. Available: https://en.wikipedia.org/wiki/Python_(programming_language).

[6] D. S. Foundation, «Django,» 2005-2018. [En línea]. Available: https://www.djangoproject.com/ .

[7] D. b. B. M. Picamán, «Oficina de Software Libre de la Universidad de Granada,» 2018. [En línea]. Available: https://osl.ugr.es/2016/10/17/entornos-virtuales-en-python-con-virtualenv/.

[8] @. a. @fat, «Boostrap,» [En línea]. Available: https://getbootstrap.com/.

[9] T. j. Foundation, «jquery.com,» 2018. [En línea]. Available: https://jquery.com/.

[10] C. M. Ricardo, BASE DE DATOS, MEXICO: McWRAW-HILL, 2004.