UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escuela Técnica Superior de Ingeniería Informática
Universidad Politécnica de Valencia

Desarrollo de una aplicación web para un gimnasio en Django

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Álvaro Guerrero Sánchez

Tutor: José Vicente Busquets Mataix

Curso 2021-2022

Resumen

El proyecto a desarrollar es una aplicación web para un gimnasio, desarrollada en el framework Django.

Los usuarios de la aplicación podrán registrarse y autenticarse, y consultar desde la misma las actividades ofertadas por el centro, los productos que tengan a la venta y también podrán enviar correos de consulta a los propietarios del centro.

Por otro lado, los propietarios del gimnasio, que tendrán permisos de administrador, también podrán gestionar las actividades y los productos que estén en la aplicación y enviar correos de spam a los usuarios que estén registrados en la aplicación.

Palabras clave: Django, Python, Aplicación Web, Entorno de Desarrollo, Andamiaje

Abstract

The project to be developed is a gym's web application, developed in the framework called Django.

The users of the application will be able to register, authenticate and consult the activities offered by the center, the products they have for sale, and they will also be able to send request emails to the managers of the center.

On the other hand, gym managers, who will have administrator permissions, will also be able to manage the activities and products that are in the application and send spam emails to users who are registered in the application.

Key words: Django, Python, Web Application, Framework, Scaffolding

Índice

1. Introducción

1.1. Introducción

A continuación expondremos los objetivos y motivaciones, por las cuales ha tenido lugar el desarrollo del proyecto de este Trabajo de Fin de Grado.

1.1.1. Motivación

La idea de desarrollar nuestra aplicación web, surge a raíz de la necesidad de gestionar, tanto actividades como productos en venta, de notificar a los usuarios de cualquier información relevante y de responder dudas de los usuarios de la aplicación, por parte de los trabajadores de un centro deportivo. Las actividades mencionadas anteriormente serían las que podrían realizar.

1.1.2. Motivación personal

En mi caso, el desarrollo de esta aplicación web surge por petición de los dueños del centro deportivo, que mencioné en el apartado anterior, y la decisión de desarrollar dicha aplicación en el entorno de programación escogido (Django), tuvo lugar porque en la empresa que realicé las prácticas, estaban barajando la posibilidad de empezar a hacer uso de este entorno. Estos motivos, me ayudaron a decidir tanto el tema del proyecto, como el entorno de programación.

1.1.3. Objetivo

El objetivo principal del proyecto es la creación de una aplicación que ayude a los propietarios de un centro deportivo, a gestionar tanto las actividades que ofertan como los productos que tienen a la venta, entendiendo por gestionar la creación, modificación o eliminación de cualquiera de los dos tipos de entrada. De este objetivo principal, parten otros, como el de que mediante la aplicación se puedan dar a conocer las actividades que se están impartiendo o los productos que se están vendiendo en ese momento, o la posibilidad de poder responder las dudas que consulten los usuarios haciendo uso de la aplicación, o el de notificar a todos los usuarios registrados con cualquier tipo de información, que los trabajadores crean conveniente notificar.

1.2. Django

Django es un entorno de programación en Python diseñado para el desarrollo de aplicaciones web. Es un framework sencillo de instalar, con posee una documentación muy completa y extensa, y que viene con un Sistema Gestor de Bases de Datos ya incorporado (sqlite3), a pesar de que no es el único que puede utilizar, el funcionamiento desde la aplicación es el mismo en cualquiera, ya q ue emplea un sistema de modelos que indica a los Gestores lo que deben hacer. En este momento, el entorno emplea python3, por lo que al ir actualizando la versión del lenguaje de programación, los métodos y librerías que se utilicen no se quedan obsoletos. Esto ayudará a la hora de tener que realizar futuros mantenimientos en las aplicaciones desarrolladas en Django[1].

Algunos de los puntos fuertes de este entorno son:

• El uso de un sistema de modelos de bases de datos, que hace que el uso de los Gestores de Bases de Datos sea independiente de la aplicación, ya que el uso se realiza mediante código en la aplicación.

• La posibilidad de dividir la aplicación, en aplicaciones con funcionalidades pequeñas que luego se junten en una, permitiendo darle al código un aspecto más conciso, ya que cada aplicación se encargará de un requisito de la aplicación.

• La capacidad de importar modelos de bases de datos, formularios y URLs entre aplicaciones.

• Se puede importar Bootstrap[2], para el diseño de la página web.

• Permite el uso de andamiaje o scaffolding, que consiste en usar plantillas de código predefinidas, para determinadas situaciones, para generar el código final mediante el que el programador podrá especificar cómo se usa la base de datos, es decir, permite leer, crear, modificar y eliminar entradas de la base de datos.

Existen bastantes aplicaciones web que utilizan Django, como mínimo en alguno de sus componentes:

• Instagram.

• Pinterest.

• Nasa Science.

• National Geographic.

• The New York Times.

2. Especificación de requisitos

2.1. Introducción

Esta especificación de requisitos, se redacta con el propósito de ayudar a futuros desarrolladores a la hora de programar, correctamente el producto, siguiendo todas las especificaciones marcadas por el cliente. Así como también sirve para, de la forma más detallada posible, describir las interfaces, el software y los requerimientos del cliente.

2.1.1. Propósito

Ayudar a los usuarios a llegar a acuerdos para establecer la base de lo que será el desarrollo del producto, y a los usuarios finales del producto a comprender lo que el cliente desea.

2.1.2. Ámbito del sistema

Nombre del producto a desarrollar: Muscle Gym. Funciones del producto:

-Permitir la creación, modificación y eliminación de actividades y productos.

-Realizar consultas de los usuarios a los clientes de la aplicación.

-Dar a conocer a los usuarios los productos y actividades introducidos por los clientes vía aplicación.

El principal objetivo de la aplicación es mejorar la comunicación del cliente con los usuarios, gracias a la consulta de dudas, y la posibilidad de los usuarios de ver las actualizaciones de las actividades y de la tienda.

2.1.3. Definiciones, acrónimos y abreviaturas

Definiciones:

-Almacenamiento: Capacidad de un sistema informático de guardar información, mediante cualquier tipo de dispositivo.

-Actualizar: Consiste en la adición, eliminación o modificación de cualquier tipo de información o registro dentro de la aplicación.

-Base de datos: Conjunto de datos pertenecientes a un mismo contexto, que son almacenados, de forma organizada, en la memoria de un ordenador, para su posterior uso. Las bases de datos están divididas en campos y registros para el almacenamiento de información, los campos hacen referencia a los atributos y los registros hacen referencia a la información de cada uno de esos atributos.

-Botón: Objeto de la interfaz, que permite la interacción del usuario, realizando un evento tras presionarlo.

-Interfaz: Medio de conexión que permite a los usuarios comunicarse con el sistema.

-Internet: Conjunto de redes de comunicaciones interconectadas, que permiten la conexión y comunicación entre los distintos ordenadores conectados, es decir, un ordenador conectado a una red puede conectarse a cualquier otro ordenador de la red.

-Login: Proceso de autenticación de un determinado usuario mediante la identificación de unas determinadas credenciales.

-Framework: Marco de trabajo, habitualmente utilizado por programadores, para desarrollar software. El hecho de utilizar un framework puede ser muy beneficioso, ya que ayuda a evitar tener que escribir código de manera repetitiva, lo cual hace que el código sea más consistente.

-Sistema Gestor de Bases de Datos: Es un conjunto de programas que nos permite realizar las funciones de extraer, modificar y guardar información en una base de datos, o lo que es lo mismo, realizar las funciones de las bases de datos, pero además también tienen herramientas que realizan funciones, que nos permiten analizar los datos que se encuentran en las bases de datos.

Acrónimos:

-URL: Acrónimo de Uniform Resource Locator, es la dirección única y concreta que tiene cada una de las páginas de la web, es decir, ayudan a los usuarios en la localización de las mismas.

Abreviaturas:

-SW: Software.

-HW: Hardware.

-Ing.: Ingeniero/a.

2.1.4. Referencias

- IEEE Standard 830-1998.
- https://docs.djangoproject.com/en/3.2/
- Wikipedia.
- https://riunet.upv.es/bitstream/handle/10251/55569/G%c3%93MEZ%20-%20Aplicaci%c3%b3n%20Web%20de%20bases%20de%20datos%20en%20PHP%20usando%20el%20Framework%20Symfony.pdf?sequence=1&isAllowed=y
- https://www.inf.upv.es/www/etsinf/wp-content/uploads/2019/03/EstructurayContenidodo3_19.pdf
- https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf
- Learning Python: Powerful Object-Oriented Programming del autor Mark A. Lutz.

2.1.5. Visión general del documento

El documento de especificación de requisitos está dividido en tres partes:

-Introducción: En esta parte del documento es donde se detallan todos los objetivos, tanto del documento como del producto en cuestión.

-Descripción general: Describe, de forma general, una perspectiva del producto que se quiere desarrollar, así como también se describen las características del usuario y las limitaciones que pudiera tener.

-Requisitos específicos: Muestra detalladamente todos los requisitos deseados por el usuario en el producto final.

2.2. Descripción general

2.2.1. Perspectiva del producto

El producto a desarrollar es un sistema totalmente independiente de otros sistemas, dividido en varios módulos, que serían el módulo de gestión de usuarios, el de gestión de actividades, el de gestión de la tienda y el de contacto; y que cada uno se encargará de un ámbito dentro de la aplicación.

2.2.2. Funciones del producto

La aplicación web a desarrollar tiene diversas funcionalidades:

-Tiene una pestaña de actividades, que mostrará cada una de las distintas actividades ofrecidas por el centro, con una descripción de las mismas y los usuarios tendrán la posibilidad de saber el día y la hora en el que se realizan cada una de las actividades que les interesen.

-Tiene una pestaña de tienda, en la que los usuarios podrán mirar cada uno de los productos puestos a la venta por el centro, pudiendo ver tanto la imagen del mismo, como el precio de cada uno de los productos. Tiene una pestaña de contacto, para poder preguntar vía gmail cualquier duda que tengan, cada una de las consultas serán enviadas a una misma dirección de correo, para que el centro pueda responder las consultas desde esa dirección.

-Tiene una pestaña de registro de usuario, desde la cual se podrán registrar los usuarios que quieran, siempre y cuando no estén ya registrados. Una vez realizado el registro, la aplicación realiza automáticamente el login del usuario, para que una vez registrado no tenga que logearse también.

-Tiene una pestaña de acceso a usuario, cualquier usuario que ya se haya registrado, podrá acceder utilizando el nombre de usuario y contraseña con los que se haya registrado, de esta manera podrá acceder a algunas funcionalidades a las que solo pueden acceder los usuarios registrados en la aplicación.

-Siempre que el usuario esté logeado en la aplicación, aparecerá un botón de cierre de sesión.

-Tiene una pestaña para crear actividades, desde la cual se podrán crear todas las actividades que se el centro vaya añadiendo, esta funcionalidad solo podrá ser utilizada por aquellos usuarios logeados que tengan permisos de administrador.

-Tiene una pestaña de modificar y eliminar actividades, todas las actividades que el centro deje de ofertar podrán ser eliminadas inmediatamente desde la propia aplicación web, de la misma forma, cualquier actividad que quiera ser modificada, también podrá serlo, el usuario será redirigido a la misma ventana que para crear las actividades, solo que con los datos de la actividad correspondiente rellenados. A esta funcionalidad solo podrán acceder aquellos usuarios registrados que tengan permisos de administrador.

-Tiene una pestaña de creación de productos, desde la que se podrán añadir todos los productos de la tienda que el centro vaya añadiendo al stock, solo podrán utilizar esta funcionalidad aquellos usuarios registrados que tengan permisos de administrador.

-Tiene una pestaña de modificación y eliminación de productos, todos los productos que el centro deje de tener en stock podrán ser eliminados directamente desde la misma aplicación, de la misma forma, también podrán ser modificados los productos, el usuario será redirigido a la misma ventana de creación de productos, solo que con los datos rellenados. Solo podrán hacer uso de esta funcionalidad aquellos usuarios registrados que tengan permisos de administrador.

2.2.3. Características de los usuarios

Los usuarios de esta aplicación podríamos dividirlos en tres grupos:

-El primer grupo estaría formado por los usuarios casuales de la aplicación, es decir, personas que harían uso de la aplicación buscando información, para preguntar dudas o para ver los productos que están a la venta en la tienda. No haría falta ningún tipo de conocimiento, nada más que tener unas nociones básicas sobre el funcionamiento de las aplicaciones web, en general.

-El segundo grupo estaría formado por usuarios que ya hubieran realizado un registro en la aplicación web, de manera que podrían realizar las mismas funciones que los usuarios casuales, solo que también podrían realizar compras en la tienda, cosa que los usuarios sin registro no podrían hacer. De la misma manera que los usuarios casuales, los usuarios que ya se hubieran registrado ,no haría falta que tuvieran ningún tipo de conocimiento concreto, más allá que tener unas nociones mínimas sobre el funcionamiento de las aplicaciones web.

-El tercer grupo de usuarios estaría compuesto por el personal del gimnasio, más concretamente, aquellos usuarios que tienen permisos de administrador, lo cual les permitiría hacer uso de una serie de funcionalidades, que el resto de usuarios no podrían realizar, como añadir y eliminar actividades del centro, y añadir y eliminar productos de la tienda. Tampoco haría falta que estos usuarios tuvieran grandes conocimientos, más allá que tener un nivel básico a la hora de utilizar una aplicación web, pero sí que deberán tener conocimientos acerca del funcionamiento y gestión del centro, ya que una de las funciones que deberán realizar es la de contestar las dudas y preguntas que los usuarios les envíen por correo, y para ello tendrán que saber acerca del funcionamiento del centro, para poder responderlas correctamente.

2.2.4. Restricciones

El proyecto se realizará en Django, un framework de código abierto que se utiliza para desarrollar aplicaciones web escritas en Python, en este proyecto se utilizará Python, lenguaje orientado a objetos, para desarrollar el código que supondrá la capa lógica de la aplicación, HTML para definir la estructura y el diseño de la aplicación web, y CSS para el diseño gráfico de la aplicación.

El Sistema Gestor de Bases de Datos (SGBD) que se empleará se llama SQLite3, ya que es el gestor que utiliza Django por defecto. También se utilizará Bootstrap, que es una biblioteca que se utilizará también para el diseño de la aplicación.

Durante el desarrollo del proyecto, se tendrán que realizar reuniones con el cliente para que proponga mejoras o alternativas mientras la aplicación todavía se encuentre en fase de desarrollo, de esta forma, se tratarán de evitar cambios estructurales de la aplicación que sean muy complejos de realizar, una vez que la aplicación esté más avanzada. De la misma forma, para el desarrollo del proyecto, se usará la metodología Scrum, planificando sprints para el desarrollo de cada uno de los componentes de la aplicación, en un plazo de tiempo ya establecido con anterioridad, incluyendo en los sprints posibles retrasos también.

2.2.5. Suposiciones y dependencias

No se ha encontrado ningún factor, que en el caso de que cambiara, provocaría un cambio en los requisitos.

2.2.6. Requisitos Futuros

Que la aplicación fuera tan utilizada que fuera necesario hacer uso de una base de datos distribuida.

2.3. Requisitos específicos

-R1: Permitir la autenticación de usuarios.

-R2: Permitir el registro de usuarios.

-R3: Permitir el cierre de sesión a los usuarios autenticados.

-R4: Permitir a los usuarios consultar las actividades.

-R5: Permitir a los usuarios consultar los productos de la tienda.

-R6: Permitir la adición de actividades.

-R7: Permitir la modificación de actividades.

-R8: Permitir la eliminación de actividades.

-R9: Permitir la adición de productos de la tienda.

-R10: Permitir la modificación de productos de la tienda.

-R11: Permitir la eliminación de productos de la tienda.

-R12: Permitir el envío de correos de consulta.

-R13: Permitir el envío de correos spam.

2.3.1. Interfaces externas

2.3.1.1 Interfaces de usuario

En cuanto a las interfaces de usuario, se refieren a aquellas ventanas que los usuarios deben manipular, para poder realizar cada una de las distintas funcionalidades que poseerá la aplicación. En el caso de nuestra aplicación, estas interacciones las realizará el usuario usando, únicamente, el teclado y el ratón. Las interfaces que ayudarán al usuario final a interactuar con la aplicación serán, básicamente:

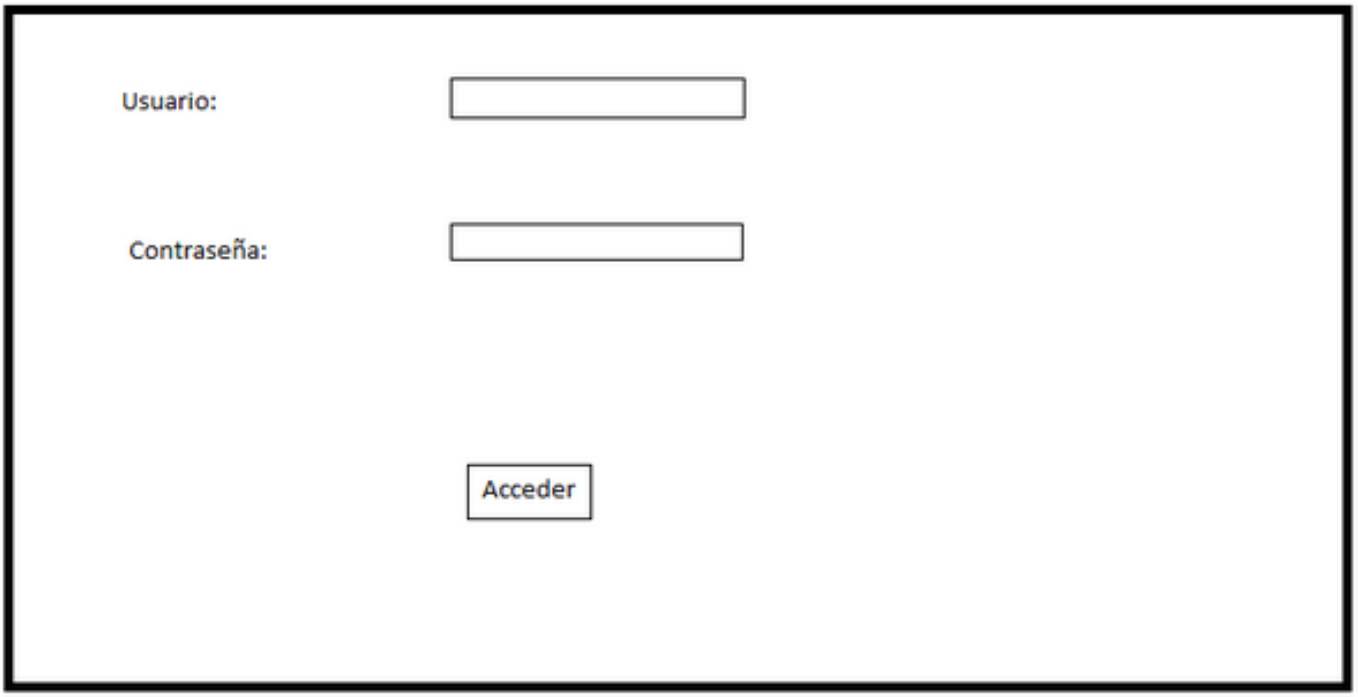
-Los botones.

-Cuadros de diálogo.

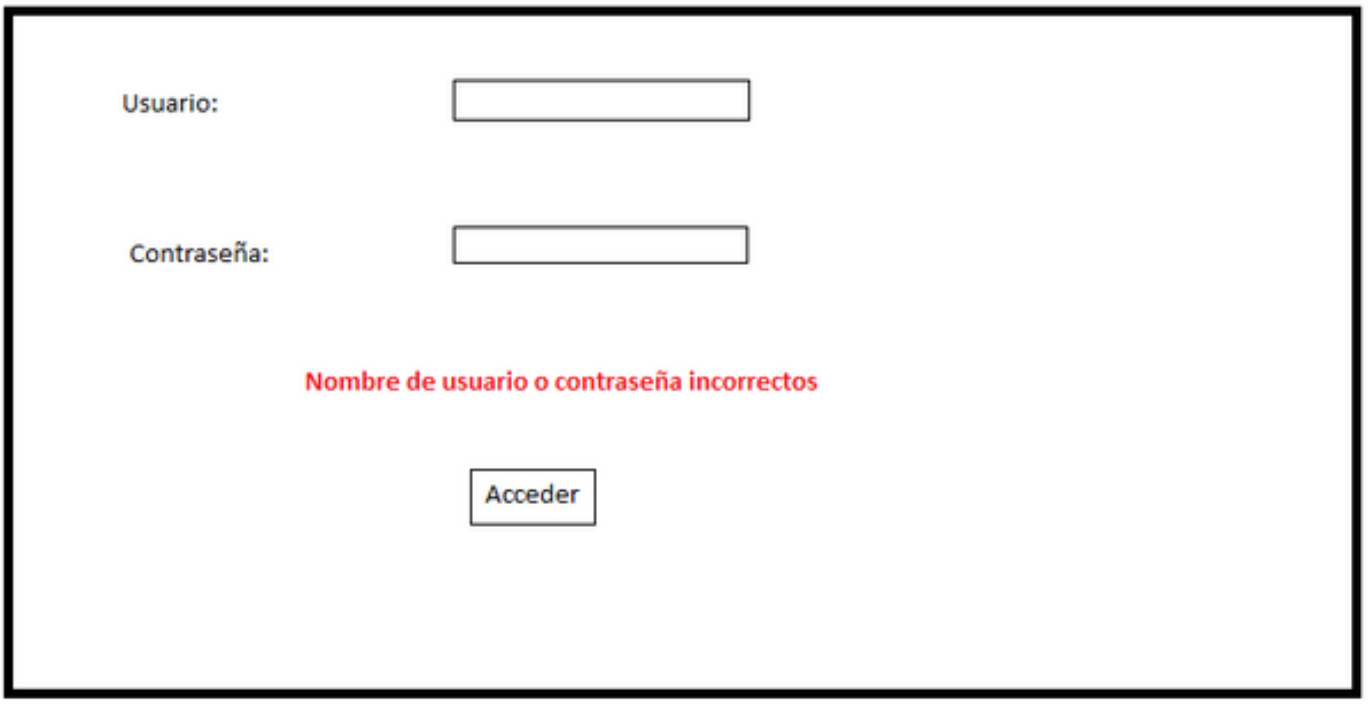
-Formularios que se utilizarán para la adición y eliminación, tanto de actividades como de productos.

-Mensajes que aparecen en la aplicación para guiar al usuario en el correcto uso de las funciones que quiera realizar.

Un ejemplo de interfaz de usuario, que debería aparecer en la aplicación, es el de autenticación de los usuarios, ejemplo:



Si el usuario introdujera de forma incorrecta sus credenciales, el nombre de usuario o la contraseña, debería aparecerle un mensaje de error, diciendo que los datos que ha introducido son incorrectos, como es el caso de la siguiente imagen:



2.3.1.2 Interfaces de hardware

En cuanto a las interfaces hardware, se hace referencia al conjunto de dispositivos que facilitan al usuario de una aplicación,intercambiar información con el ordenador, ya sea leyendo los datos o introduciendolos. Las interfaces hardware que requerirá nuestra aplicación serán:

-El monitor de pantalla: Nuestra aplicación deberá mostrar información con el usuario, mediante la pantalla del ordenador.

-El ratón: La aplicación deberá mostrar por pantalla tanto los movimientos, como cada uno de los cliqueos que realicemos con el ratón. El usuario podrá hacer uso del ratón para hacer clic tanto en botones como en cuadros de diálogo. El ratón será el dispositivo de entrada de datos con el cual el usuario podrá poner el cursor en los cuadros de texto, o desplazarse por las distintas funcionalidades de la aplicación.

-El teclado: Nuestra aplicación deberá interactuar con cada una de las pulsaciones que el usuario realice en el teclado, siempre y cuando el usuario haya puesto el cursor en un cuadro de diálogo. El teclado será el dispositivo de entrada de datos con el que el usuario introducirá información y consultará dudas.

2.3.1.3 Interfaces de software

En cuanto a las interfaces de software, son aquellas que permiten a los usuarios obtener información sobre cualquier proceso o herramienta de control que esté ejecutándose en ese momento, mediante elementos que el usuario puede observar normalmente en la pantalla del ordenador.

En el caso de nuestra aplicación, no posee ningún tipo de elemento que informe al usuario del estado de ningún proceso, debido a que no posee funcionalidades que sean tan pesadas computacionalmente, como para durar tanto tiempo que sea necesario informar al usuario de su estado.

2.3.1.4 Interfaces de comunicación

Respecto a las interfaces de comunicación, hace referencia a aquellos elementos que permiten la transmisión de información, de un equipo de datos a un módem, es decir, un dispositivos que convierte señales, con el fin de permitir la comunicación entre ordenadores a través de una línea; o hacia el medio que se usará como transmisor para comunicar con otro equipo de datos.

La interfaz de comunicación entre nuestra base de datos y nuestra aplicación desarrollada en Python, la realizará el Sistema de Gestión de Bases de Datos que usaremos por defecto, el SQLite3.

2.3.2. Funciones

2.3.2.1 Requisito Funcional 1

El primer requisito funcional a desarrollar será el de permitir la autenticación[3] de los usuarios. Nuestra aplicación deberá permitir a los usuarios autenticarse mediante un nombre de usuario y una contraseña, para poder hacer uso de algunas funcionalidades específicas a las que solo podrán acceder los usuarios registrados o aquellos usuarios que tengan permisos de administrador.

Los campos de entrada que los usuarios deberán rellenar para autenticarse serán:

-username.

-password.

Cuando los usuarios accedan a la ventana de autenticación, se encontrarán con dos campos a rellenar el de nombre de usuario y el de contraseña, si los rellena correctamente será redirigido a la página principal, como usuario registrado, de manera que le aparecerá un mensaje con su nombre de usuario. En el caso de que el usuario sea administrador le pondrá que es admin.

En el caso de que el usuario haya cometido un error al intentar autenticarse, se le notificará mediante un mensaje:

-Si el usuario no ha rellenado alguno de los dos campos, el sistema deberá informarle de que ambos campos deben estar rellenados, para poder realizar el acceso del usuario correctamente.

-Si el usuario introduce un nombre de usuario o contraseña, que no coincidan con el de algún usuario que no esté registrado en la base de datos, se le deberá notificar al usuario que o bien el usuario o bien la contraseña que ha introducido no coincide con ninguno de los usuarios registrados en la base de datos.

Este requisito va dirigido a los usuarios registrados en la aplicación y a los que tendrán los permisos de administrador, o sea, los empleados del gimnasio.

2.3.2.2 Requisito Funcional 2

El segundo requisito funcional que deberá cumplir nuestra aplicación, será el de permitir el registro de los nuevos usuarios. Nuestra aplicación deberá permitir que cualquier usuario que quiera realizar un registro en el producto, siempre y cuando no sea un usuario ya registrado, pueda realizar el registro perfectamente. De esta manera, el nuevo usuario registrado podrá hacer uso de funcionalidades que antes no podía utilizar. Los campos de entrada que los usuario deberán rellenar para realizar el registro serán:

-Nombre.

-Apellido.

-Correo electrónico (Gmail).

-Nombre de usuario.

-Contraseña.

-Contraseña de confirmación.

Cuando los usuarios accedan a la ventana de registro se encontrarán los campos mencionados anteriormente, una vez los haya rellenado todos, si el registro se ha realizado correctamente, la aplicación deberá realizar automáticamente el proceso de autenticación de dicho usuario, de manera que el nuevo usuario registrado será redirigido a la ventana de inicio de la aplicación, como usuario registrado.

En el caso de que haya habido algún error en el registro, el usuario deberá recibir un mensaje de error:

-A pesar de que en la ventana de registro deberán aparecer algunas directrices, como el tipo y número de caracteres aceptados por la aplicación, en el caso de que el usuario introduzca algún carácter incorrecto, deberá recibir un mensaje informando del error que impide el registro.

-Si el usuario no rellena alguno de los campos del registro, la aplicación deberá notificarle que los campos deben ser rellenados para que el proceso de registro se realice correctamente.

-En caso de que un usuario intente registrarse, y utilice un nombre de usuario, un correo electrónico o una contraseña, que ya posea alguno de los usuarios ya registrados, deberá ser informado del motivo por el cual no ha podido realizarse correctamente el proceso de registro.

Este requisito va dirigido a aquellos usuario que todavía no se han registrado, es decir, usuarios casuales que se han interesado en la aplicación o nuevos trabajadores del centro que necesitan registrarse con permisos de administrador, para poder desempeñar su trabajo mediante funcionalidades de la aplicación.

2.3.2.3 Requisito Funcional 3

El tercer requisito funcional que deberá garantizar nuestra aplicación, será el de permitir a los usuarios registrados que hayan realizado una autenticación, cerrar la sesión de la cuenta con la que hayan hecho el acceso.

La aplicación deberá permitir a los usuarios autenticados que, mediante un botón, puedan cerrar la sesión de su cuenta. Para ello, el usuario deberá haberse autenticado previamente, y entonces le deberá aparecer el botón que le permita realizar el cierre de sesión.

En el caso de esta funcionalidad, los usuarios no deberán rellenar ningún campo de entrada, ya que solo tendrán que pulsar el botón que les permitirá cerrar sesión, o sea, realizar la funcionalidad. Este requisito va dirigido a aquellos usuarios que se hayan registrado, ya sean usuarios registrados o con permisos de administrador, y que quieran realizar un cierre de sesión de su cuenta.

2.3.2.4 Requisito Funcional 4

El cuarto requisito funcional que deberá cumplir nuestra aplicación, deberá ser el de permitir a los usuarios consultar las actividades, que estén en ese momento dadas de alta en la aplicación.

La aplicación deberá garantizar que todos aquellos usuarios, que quieran consultar las actividades ofertadas por el centro, puedan acceder a la ventana correspondiente sin ningún problema.

En el caso de esta funcionalidad, no será necesario que el usuario rellene ningún campo de entrada, ya que simplemente tendrá que seleccionar el botón que le permita acceder a la ventana que le permitirá consultar las actividades.

Esta funcionalidad va dirigida a todos los usuarios, ya sean usuarios casuales, usuarios registrados o usuarios registrados que posean permisos de administrador, ya que cualquiera de los tres tipos de usuarios podrá tener la necesidad de consultar las actividades de la aplicación.

2.3.2.5 Requisito Funcional 5

El quinto requisito funcional que deberá permitir nuestra aplicación, será el de permitir a los usuarios consultar los productos que se encuentren a la venta en la ventana de tienda de la aplicación.

La aplicación deberá garantizar, que cualquier usuario pueda consultar, sin ningún tipo de problema, los productos de la tienda, permitiéndoles acceder a la ventana correspondiente al apartado tienda.

En el caso de esta funcionalidad, no será necesario que el usuario rellene ningún campo de entrada, ya que simplemente tendrá que seleccionar el botón que le redirigirá a la ventana que le permitirá consultar las productos de la tienda.

Esta funcionalidad va dirigida a todos los usuarios, ya sean usuarios casuales, usuarios registrados o usuarios registrados que posean permisos de administrador, ya que cualquiera de los tres tipos de usuarios podrá tener la necesidad de consultar los productos de la aplicación.

2.3.2.6 Requisito Funcional 6

El sexto requisito que deberá cumplir la aplicación, será el de permitir a los usuarios registrados con permisos de administrador, añadir actividades ofertadas por el gimnasio en la aplicación.

La aplicación deberá garantizar a los usuarios con permisos, poder añadir actividades ofertadas por el gimnasio. Evidentemente, a estos usuarios se les tienen que haber dado permisos de administrador.

Los campos de entrada que los administradores deberán rellenar para poder crear una nueva actividad serán:

-El nombre de la actividad.

-Una descripción de la actividad.

-Una imagen sobre la actividad.

-El día en el que se impartirán las clases de dicha actividad.

-La hora de inicio de la actividad.

-La hora de finalización de la actividad.

Cuando los usuarios con permisos de administrador accedan a la ventana de crear una nueva actividad, deberán rellenar los campos mencionados anteriormente, para poder efectuar la adición de la actividad correctamente. Una vez se haya realizado la creación de la actividad, al usuario le aparecerá un mensaje de que la actividad ha sido creada correctamente, después podrá ir a la ventana de actividades, de manera que podrá comprobar si la nueva actividad que ha creado, ha sido nombrada y descrita correctamente, si le ha adjuntado la imagen correspondiente, y si el horario que le ha asignado es el correcto.

En el caso de que haya habido algún error cuando los usuarios con permisos de administrador intenten crear una nueva actividad, deberán ser informados mediante algún mensaje:

-Si los usuarios se han dejado algún campo por completar, deberán ser notificados de que para poder realizar correctamente la adición de una nueva actividad, todos los campos deben estar rellenados.

Este requisito va dirigido a aquellos usuarios que poseen permisos de administrador, ya que serán los únicos que podrán acceder en la aplicación a la ventana de crear una nueva actividad.

2.3.2.7 Requisito Funcional 7

El séptimo requisito funcional que deberá cumplir nuestra aplicación, deberá ser el de permitir a los usuarios registrados con permisos de administrador, modificar las actividades de la aplicación.

La aplicación deberá garantizar que los usuarios registrados tengan la posibilidad de modificar las actividades ofertadas por el centro. Estos usuarios deben haber recibido permisos de administrador previamente, ya que solo podrán acceder a esta funcionalidad los usuarios con permisos de administrador.

Los campos de entrada que los administradores deberán rellenar para poder modificar una actividad serán:

-El nombre de la actividad.

-Una descripción de la actividad.

-Una imagen sobre la actividad.

-El día en el que se impartirán las clases de dicha actividad.

-La hora de inicio de la actividad.

-La hora de finalización de la actividad.

Los usuarios que quieran modificar una actividad, deberán pasar previamente por una ventana con el listado de actividades, desde la cual podrá decidir cuál modificar. Una vez acceda a la ventana que le permitirá modificar la actividad, le aparecerán los mismos campos que para crearla, solo que rellenados con los datos de la actividad que desee modificar. En el caso de que se haya realizado la modificación correctamente, al usuario deberá aparecerle un mensaje indicándolo.

En el caso de que haya habido algún error cuando los usuarios con permisos de administrador intenten modificar una actividad, deberán ser informados mediante algún mensaje:

-Si los usuarios se han dejado algún campo por completar, deberán ser notificados de que para poder realizar correctamente la modificación de una nueva actividad, todos los campos deben estar rellenados.

Este requisito va dirigido a aquellos usuarios que poseen permisos de administrador, ya que serán los únicos que podrán acceder en la aplicación a la ventana de modificar una actividad.

2.3.2.8 Requisito Funcional 8

El octavo requisito funcional que deberá cumplir nuestra aplicación, deberá ser el de permitir a los usuarios registrados con permisos de administrador, eliminar actividades de la aplicación.

La aplicación deberá garantizar que los usuarios registrados tengan la posibilidad de eliminar actividades ofertadas por el centro. Estos usuarios deben haber recibido permisos de administrador previamente, para poder acceder a la ventana desde la que podrán eliminar actividades.

En el caso de esta funcionalidad, no será necesario que los usuarios deban rellenar ningún campo de entrada, dado que les tendrán que aparecer las actividades, por su nombre y como si fueran un listado, y cada una de las actividades tendrá un botón para poder eliminar la actividad en cuestión.

Cuando los usuarios que se hayan autenticado, y que tengan permisos de administrador, tendrán que acceder previamente a una ventana que posea un listado con las actividades, desde la cual podrá decidir cual elimina, cuando accedan a la ventana de eliminar una actividad, deberán buscar la actividad que deseen borrar y cuando la hayan encontrado, tendrán que pulsar el botón de eliminar. Cuando se haya realizado la acción de borrado de la actividad, la aplicación deberá redirigir al usuario a la ventana de actividades, desde la cual podrá verificar que la actividad correspondiente se ha eliminado correctamente. Este requisito va dirigido a aquellos usuarios que poseen permisos de administrador, ya que serán los únicos usuarios que podrán acceder en la aplicación, a la ventana de eliminar una actividad.

2.3.2.9 Requisito Funcional 9

El noveno requisito funcional que deberá cumplir nuestra aplicación, será el de permitir a los usuarios que se hayan registrado y que posean permisos de administrador, añadir productos en la ventana de tienda de la aplicación.

La aplicación deberá garantizar que los usuarios registrados tengan la posibilidad de añadir productos en el apartado de tienda de la aplicación. Estos usuarios deben haber recibido, previamente, permisos de administrador, para poder acceder a la ventana desde la que podrán añadir nuevos productos.

Los campos de entrada que los administradores deberán rellenar para poder crear un nuevo producto serán:

-El nombre del producto.

-Una imagen del producto.

-El precio del producto.

Cuando los usuarios con permisos de administrador accedan a la ventana de crear un nuevo producto en la tienda, deberán rellenar los campos mencionados anteriormente, para poder efectuar la adición del producto correctamente. Una vez se haya realizado la creación del producto, el usuario será redirigido a la ventana de la tienda, de manera que podrá comprobar si el nuevo producto que acaba de crear, ha sido nombrado correctamente, si se le ha adjuntado la imagen correspondiente, y si el precio que se le ha asignado es el correcto.

En el caso de que haya habido algún error cuando los usuarios con permisos de administrador intenten crear un nuevo producto, deberán ser informados mediante algún mensaje:

-Si los usuarios se han dejado algún campo por completar, deberán ser notificados de que para poder realizar correctamente la adición de un nuevo producto, todos los campos deben estar rellenados.

Este requisito va dirigido a aquellos usuarios que poseen permisos de administrador, ya que serán los únicos que podrán acceder en la aplicación a la ventana de crear un nuevo producto para la tienda.

2.3.2.10 Requisito Funcional 10

El décimo requisito funcional que deberá cumplir nuestra aplicación, será el de permitir a los usuarios que se hayan registrado y que posean permisos de administrador, modificar productos en la ventana de tienda de la aplicación.

La aplicación deberá garantizar que los usuarios registrados tengan la posibilidad de modificar productos en el apartado de tienda de la aplicación. Estos usuarios deben haber recibido, previamente, permisos de administrador, para poder acceder a la ventana desde la que podrán modificar productos.

Los campos de entrada que los administradores deberán rellenar para poder modificar un producto serán:

-El nombre del producto.

-Una imagen del producto.

-El precio del producto.

Cuando los usuarios con permisos de administrador accedan a la ventana de modificar un producto en la tienda,previamente tendrán que pasar por una ventana con un listado con los productos, para después seleccionar el producto a modificar, después deberán rellenar los campos mencionados anteriormente, para poder efectuar la modificación del producto correctamente. Una vez se haya realizado la modificación del producto, al usuario le aparecerá un mensaje de que el producto ha sido modificado correctamente, después podrá acceder a la ventana de tienda, de esta manera podrá comprobar si el nuevo producto que acaba de

crear, ha sido nombrado correctamente, si se le ha adjuntado la imagen correspondiente, y si el precio que se le ha asignado es el correcto.

En el caso de que haya habido algún error cuando los usuarios con permisos de administrador intenten modificar un producto, deberán ser informados mediante algún mensaje:

-Si los usuarios se han dejado algún campo por completar, deberán ser notificados de que para poder realizar correctamente la modificación de un producto, todos los campos deben estar rellenados.

Este requisito va dirigido a aquellos usuarios que poseen permisos de administrador, ya que serán los únicos que podrán acceder en la aplicación a la ventana de modificar un producto para la tienda.

2.3.2.11 Requisito Funcional 11

El decimoprimer requisito funcional que deberá cumplir nuestra aplicación, será el de permitir a los usuarios que se hayan registrado y que posean permisos de administrador, eliminar productos en la ventana de tienda de la aplicación.

La aplicación deberá garantizar que los usuarios registrados tengan la posibilidad de eliminar productos en el apartado de tienda de la aplicación. Estos usuarios deben haber recibido permisos de administrador, para poder acceder a la ventana desde la que podrán eliminar los productos de la tienda.

En el caso de esta funcionalidad, no será necesario que los usuarios deban rellenar ningún campo de entrada, dado que les tendrán que aparecer los productos, por su nombre y como si fueran un listado, y cada uno de los productos tendrá un botón para poder eliminar el producto en cuestión.

Cuando los usuarios que se hayan autenticado, y que tengan permisos de administrador, accedan a la ventana de eliminar un producto, habiendo pasado previamente por una ventana con un listado de los productos de la tienda, desde la cual podrá seleccionar el producto que quieran eliminar, después deberán buscar el producto que deseen borrar y cuando lo hayan encontrado, tendrán que pulsar el botón de eliminar. Cuando se haya realizado la acción de borrado del producto, la aplicación deberá redirigir al usuario a la ventana de tienda, desde la cual podrá verificar que el producto correspondiente se ha eliminado correctamente. Este requisito va dirigido a aquellos usuarios que poseen permisos de administrador, ya que serán los únicos que podrán acceder en la aplicación a la ventana de eliminar un producto de la tienda.

2.3.2.12 Requisito Funcional 12

El decimosegundo requisito funcional que deberá cumplir nuestra aplicación, será el de permitir el envío de correos de consulta, por parte de los usuarios registrados y no registrados, con el fin de que los trabajadores del centro les respondan cualquier tipo de dudas que les puedan surgir.

Los campos de entrada que los usuarios registrados y no registrados deberán rellenar para poder enviar un correo de consulta al centro serán:

-El nombre del usuario, que no el nombre de usuario, ya que también podrán realizar preguntas los usuarios no registrados.

-La dirección de correo electrónico (Gmail).

-La consulta que deseen realizar.

Cuando los usuarios, registrados y no registrados, deseen realizar una consulta, deberán rellenar los campos de entrada mencionados anteriormente y darle a un botón para poder enviar la consulta. En el caso de que el correo se envíe correctamente, aparecerá un mensaje de que el correo se ha enviado correctamente. Pero si ocurre algún problema a la hora de enviar el correo, se le deberá notificar al usuario mediante algún tipo de mensaje:

-Si el usuario se ha dejado algún campo de entrada por completar, deberá ser notificado de que para poder enviar el correo de consulta correctamente, deberá rellenar primero todos los campos de entrada.

-Si la aplicación tiene un problema que no le permite enviar el correo de consulta, deberá notificar al usuario que no se ha podido enviar el mensaje.

Este requisito va dirigido a aquellos usuarios registrados y no registrados, que quieran enviar un correo de consulta al centro.

2.3.2.13 Requisito Funcional 13

El decimotercer requisito funcional que la aplicación deberá cumplir, será el de permitir a los usuarios registrados que tengan permisos de administrador, el envío de correos a todos los usuarios registrados en la aplicación.

En el caso de este requisito, sólo será necesario que los usuarios con permisos de administrador tengan que rellenar un campo de entrada:

-Contenido del mensaje que les llegará a los usuarios registrados.

Cuando los usuarios registrados con permisos de administrador, quieran enviar correos a los usuarios registrados, deberán rellenar el campo con el contenido del mensaje y darle al botón de enviar, de esta manera, el mensaje escrito será enviado a todos los correos que los usuarios tendrán registrados en la base de datos.

Si se da el caso de que ocurre algún fallo a la hora de realizar esta funcionalidad, el usuario deberá recibir una notificación sobre el error que ha impedido que no se puedan enviar los correos:

-Si el usuario no rellena el área con el contenido del correo, deberá aparecerle un mensaje de que el campo debe estar rellenado para que se pueda realizar la funcionalidad correctamente.

Este requisito va dirigido a aquellos usuarios registrados que posean permisos de administrador, ya que serán los únicos que tendrán acceso a esta funcionalidad.

2.3.3. Requisitos de rendimiento

La capacidad de terminales que podrá manejar el sistema, dependerá de el servidor de base de datos que contrate el centro.

Al no ser operaciones muy complejas, computacionalmente, para el SGBD, no le debería suponer demasiado esfuerzo hacer todas las operaciones que los usuarios le hagan realizar diariamente. A pesar de eso, el servidor de base de datos contratado, deberá tener siempre un respaldo técnico para cualquier eventualidad que pueda suceder.

2.3.4. Restricciones de diseño

No tienes restricciones de diseño.

2.3.5. Atributos del sistema

La fiabilidad del sistema viene dada por el hecho de que la aplicación, garantice siempre el correcto funcionamiento de la misma.

Como la aplicación está completamente programada en código, garantiza que se le podra realizar cualquier mantenimiento y modificación en un futuro.

Gracias a que en la aplicación, tanto los lenguajes de programación empleados y la base de datos que se utiliza son compatibles con más de un sistema operativo, se garantiza la portabilidad.

La seguridad de la aplicación está garantizada, gracias al uso de contraseñas por parte de los usuarios registrados, y también por la necesidad de tener permisos de administrador para realizar algunas de las funcionalidades de la aplicación.

2.3.6. Otros requisitos

No existen otros requisitos.

3. Planificación

En este apartado de la memoria, se va a desarrollar la planificación que se ha aplicado para desarrollar el proyecto en cuestión. Para la planificación se ha seguido una metodología de desarrollo ágil llamada Scrum[4], si bien esta metodología se emplea para equipos de trabajo generalmente, en este caso, ha sido adaptada para una única persona, ya que solo una persona ha participado en el desarrollo de este proyecto.

Cabe destacar que tanto el estudio de la tecnología a emplear, como las fases de diseño y análisis del proyecto, no entran dentro de la planificación. Ya que previamente a empezar el desarrollo se le dedicó un tiempo al estudio de la tecnología, así como también se le dedicó tiempo al diálogo con el cliente, y mientras tenían lugar esas conversaciones se llevaba a cabo la fase de análisis y diseño. Salvo algún momento puntual que haya hecho falta estudiar un apartado de la tecnología para una determinada funcionalidad, o algún apunte o requisito que el cliente haya decidido añadir, los cuales se han incluido en los sprints, como tiempo que ha sobrado en los mismos.

La primera fase para realizar el proyecto en un lapso de tiempo determinado, fue realizar la planificación de sprints. Es decir, pensar y desarrollar un listado de avances que se tendrían que ir logrando a medida que los sprints se van finalizando. Como algunas de las claves para cumplir esta metodología es no crear sprints que contengan demasiadas tareas ni sobrestimar la estimación de la velocidad, desde un primer momento se procuró no cometer estos errores.

Se crearon 12 sprints, cada uno de una semana, en los cuales se estudió qué tareas introducir en cada uno. Dado que se tuvo en cuenta el hecho de que era la primera aplicación desarrollada en esta tecnología, los primeros sprints no tuvieron demasiada carga de trabajo, se basaron más en realizar el diseño de la aplicación y en las primeras páginas de la misma, las cuales no tenían código de una gran complejidad. Para el resto de sprints, se planificó que tuvieran más carga de trabajo, dado que ya había habido un periodo de práctica con la tecnología. De todas formas, los sprints no estaban planificados para tener el tiempo justo, ya que siempre tenían un margen de tiempo (horas) por si había alguna complicación de algún tipo; para, como se ha dicho anteriormente añadir alguna funcionalidad o realizar algún ajuste que el usuario haya visto conveniente, o estudiar algún apartado de la tecnología en uso para aplicar en el proyecto, como alguna funcionalidad o mejorar el diseño.

A la hora de realizar la planificación, parecía que no fuera a ser suficiente tiempo, a pesar de haber dejado el margen de horas mencionado anteriormente, por si acaso surgiera alguna complicación o hubiera que realizar una modificación. Pero después de varias semanas de cumplir los objetivos propuestos en los sprints, quedó claro que la planificación de los mismos había sido la correcta, otorgando a cada uno las horas y las tareas exactas. De hecho, el proyecto se finalizó justo en los tiempos estimados al principio de la planificación, a pesar de haber algunos sprints más intensivos que otros, lo cual prueba que la decisión de realizar la propia planificación fue una buena opción, para llevar un buen ritmo de trabajo a la hora de desarrollar la aplicación, igual que si fuera el desarrollo de una aplicación en una empresa, con unos determinados tiempos para el desarrollo de la misma.

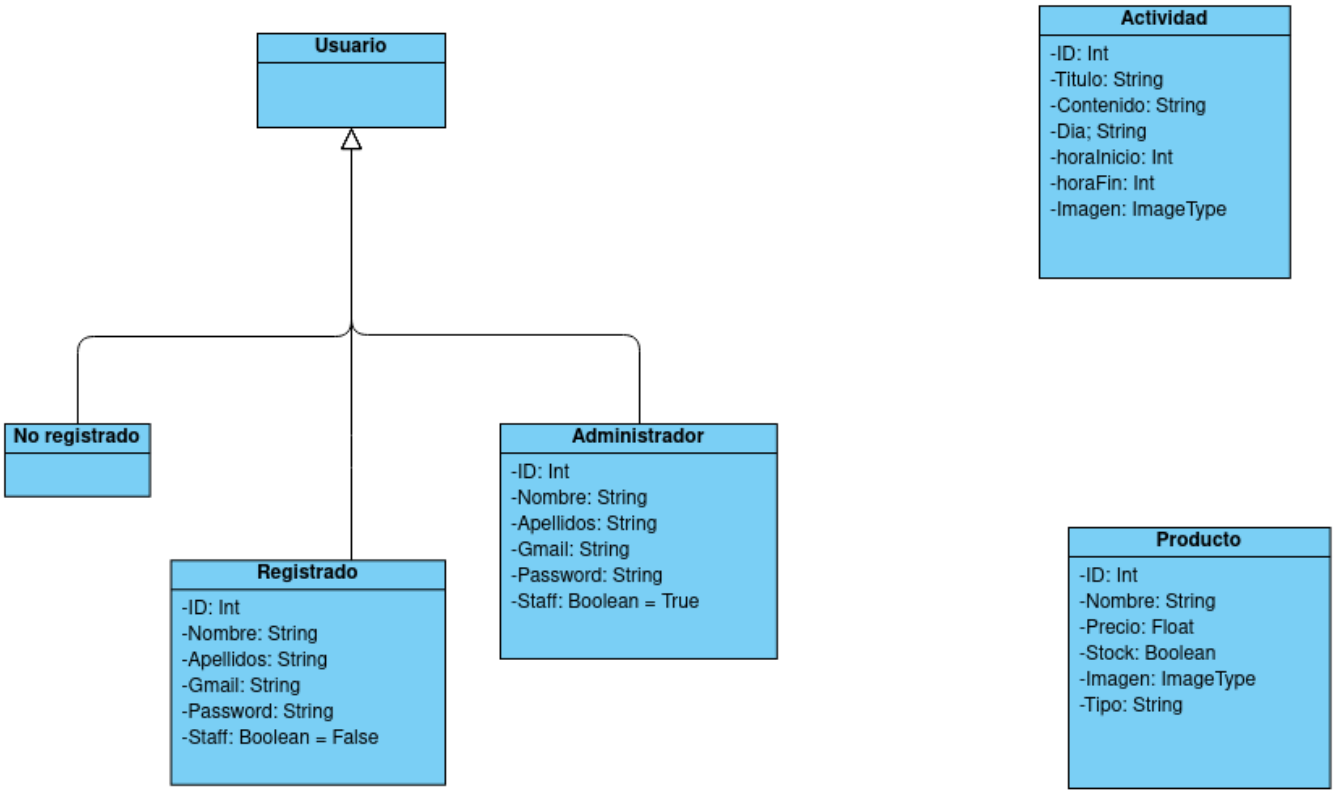
4. Análisis

4.1. Introducción

En este apartado se va a realizar un análisis que servirá para ayudar a desarrollar la futura aplicación, para este apartado se van a utilizar algunos diagramas, que nos ayudarán a realizar la etapa de diseño. Los diagramas que se van a emplear son el de clases, el de casos de uso y el de actividad.

4.2. Diagrama de clases

Para este apartado vamos a proponer el siguiente diagrama de clases, en el que podemos observar como están representados los modelos de las tablas de usuario, de actividad y de producto. Como se observa en la imagen, las tablas de producto y actividad no están relacionadas con las tablas de usuarios, eso es porque no existe ninguna relación entre ellas, a pesar de que sean los administradores los que pueden crear objetos de las dos tablas mencionadas, pero para ello solo necesitan tener los permisos correspondientes, es decir, tener el atributo staff con valor True. Al no ser necesaria una relación entre tablas, no se ve reflejada en el diagrama.

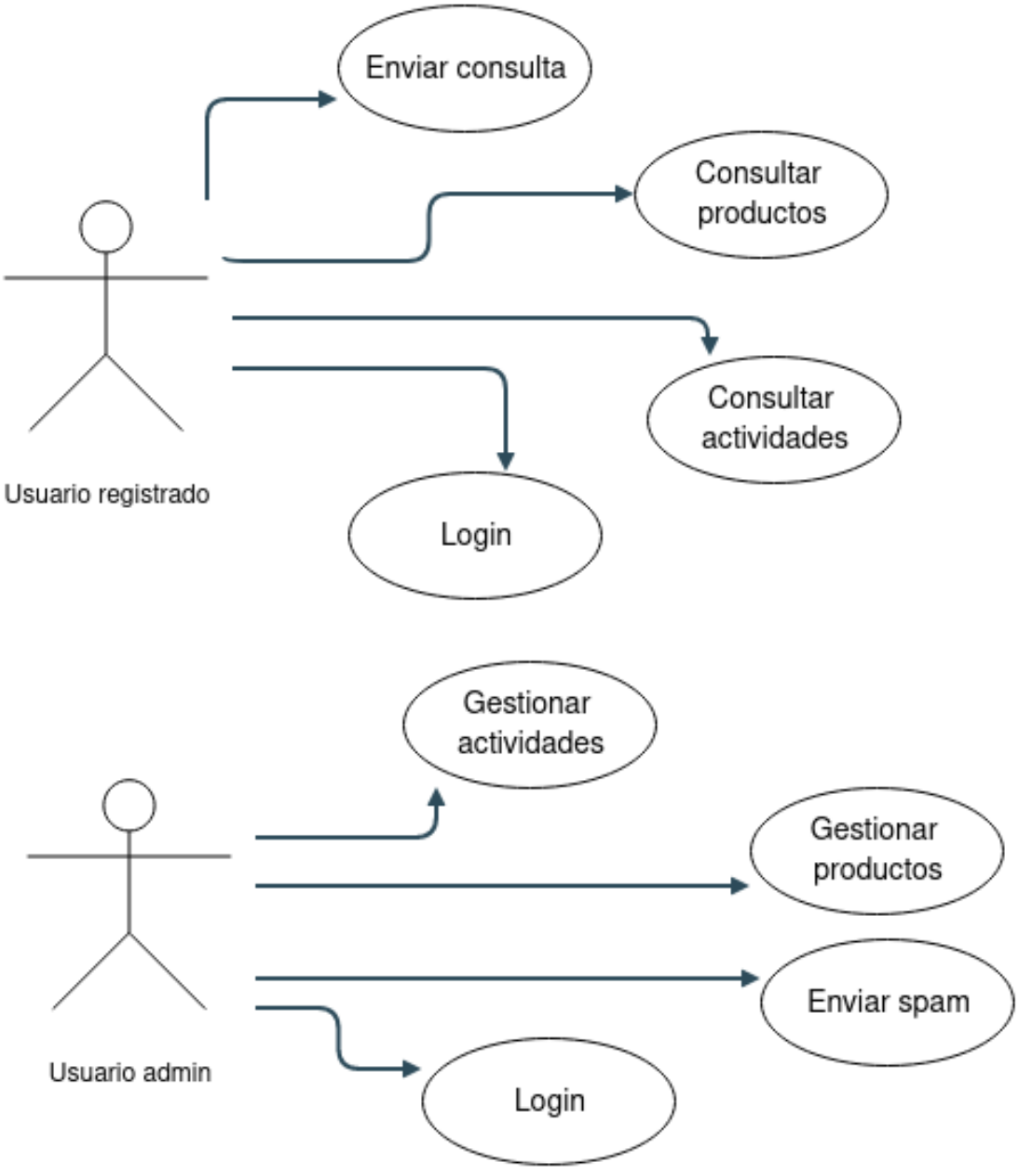


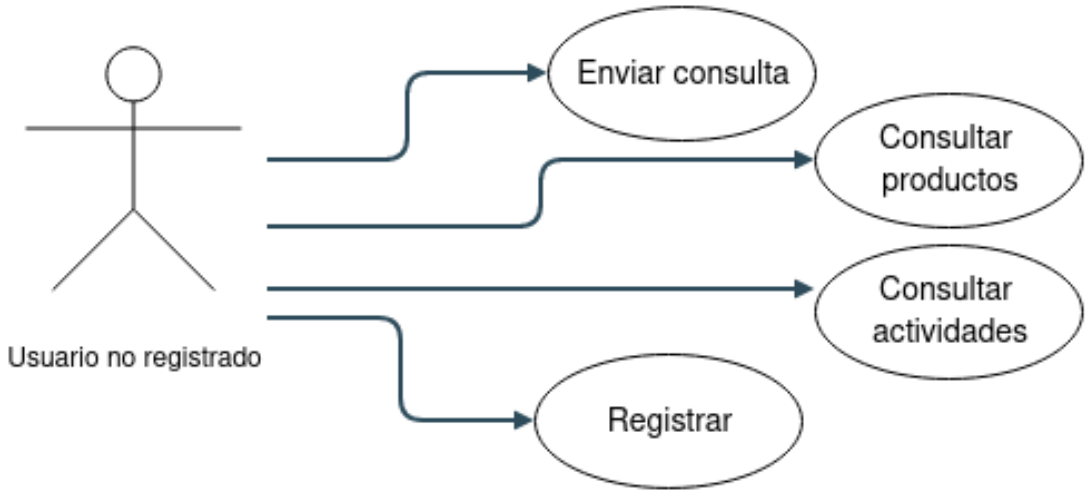
4.3. Diagrama de casos de uso

A continuación, vamos a exponer los diagramas de casos de usos de la aplicación, estos diagramas se utilizan para representar todos los procesos que puede realizar una aplicación, incluyendo el usuario de la acción y las acciones que, cada uno de los usuarios, puede tomar.

4.3.1. Caso base

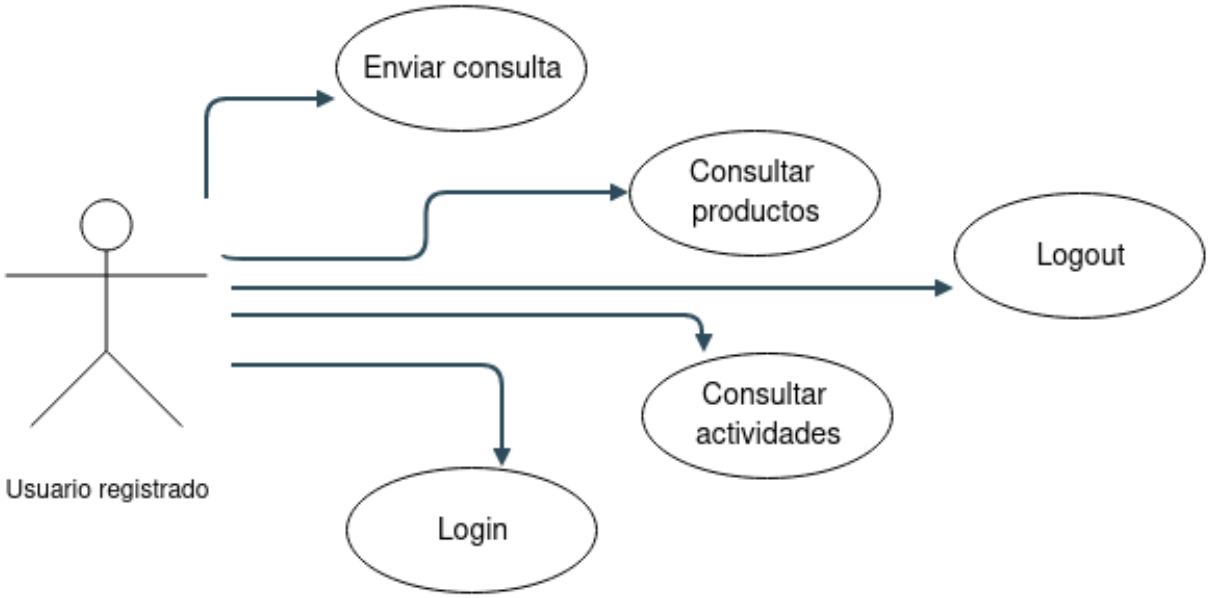
A continuación mostramos el caso de uso principal, desde el que partimos, con los tres posibles tipos de usuarios que pueden usar la aplicación.

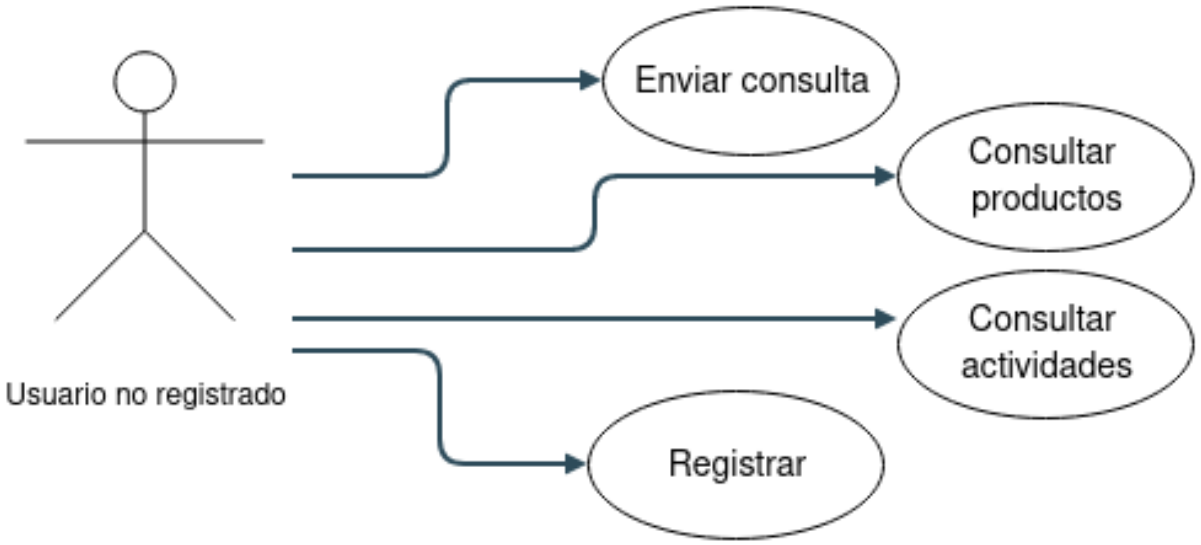


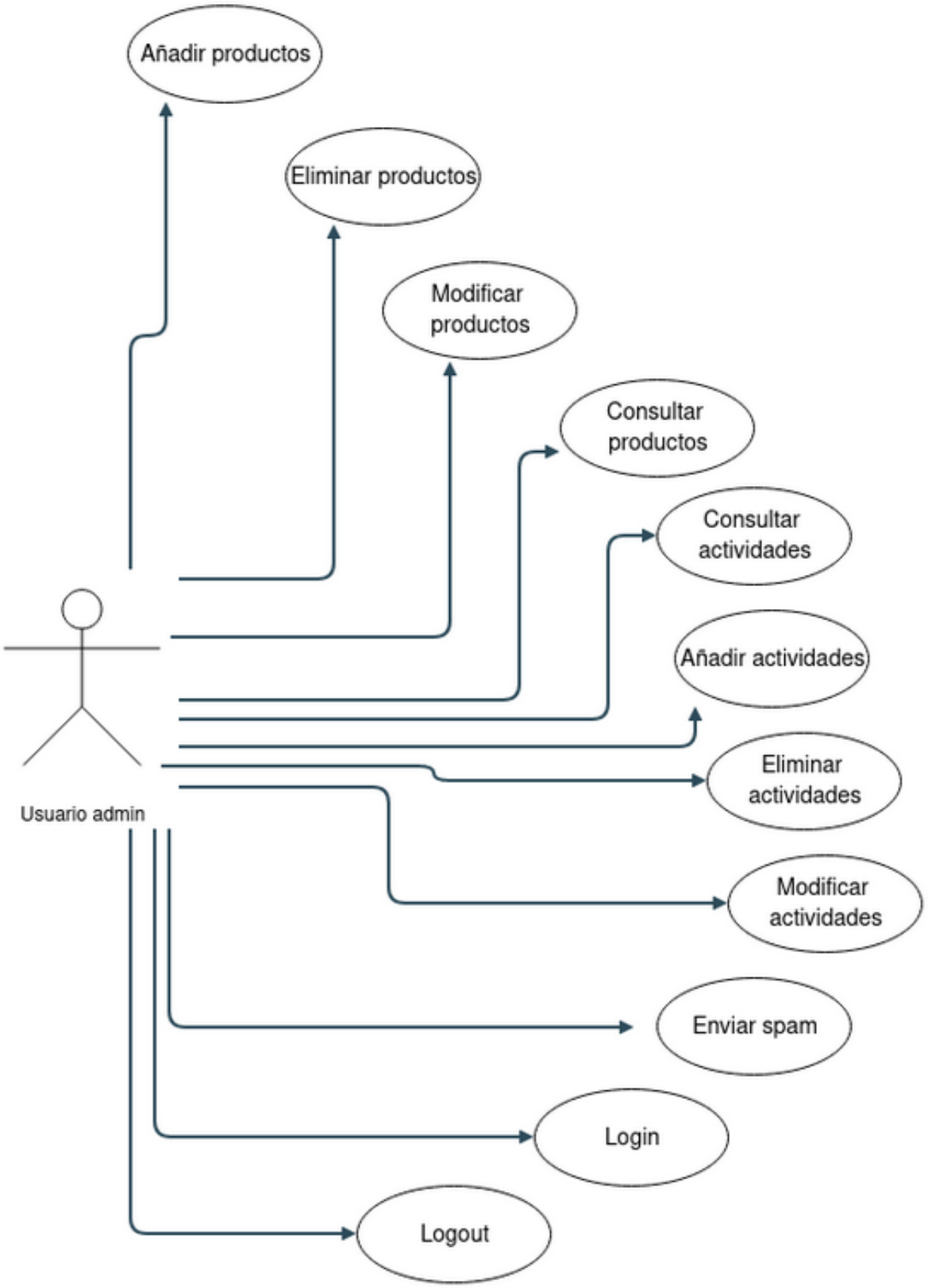


4.4. Casos de uso

A continuación mostraremos los casos de uso que muestran todas las funcionalidades que puede realizar la aplicación para los distintos tipos de usuarios.



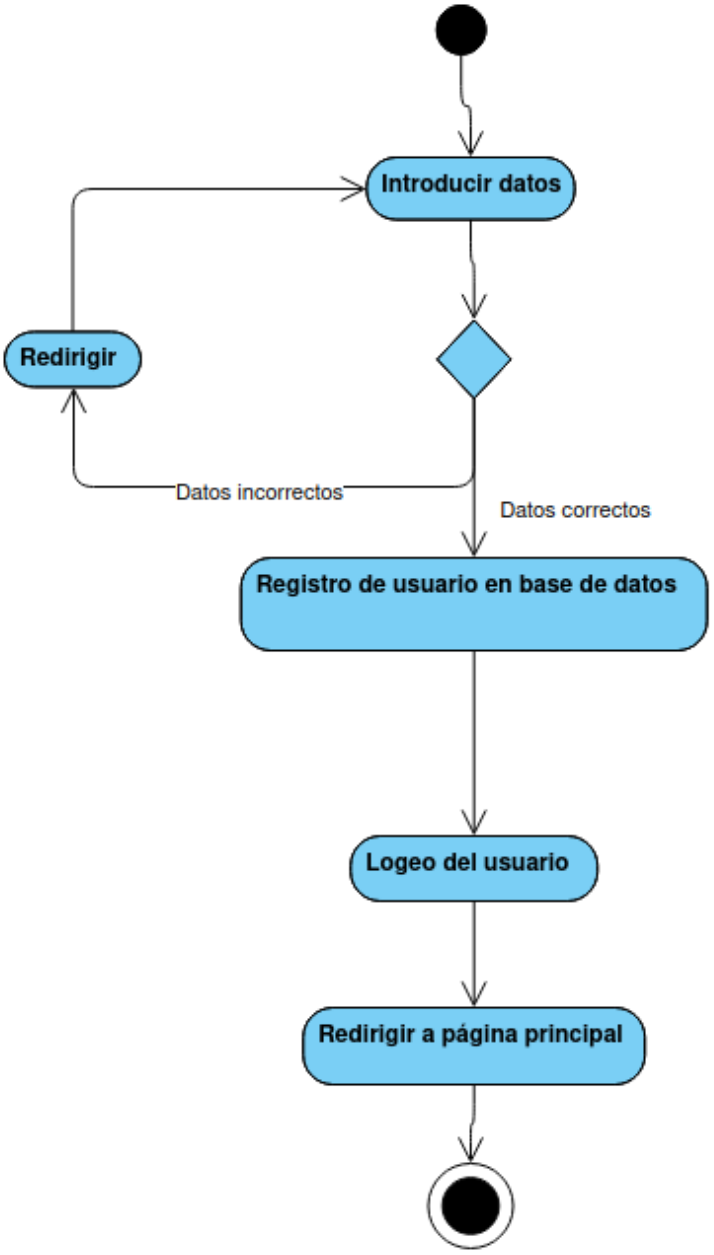

Enviar consulta
Consultar productos
Consultar actividades
Registrar
Usuario no registrado


Añadir productos
Eliminar productos
Modificar productos
Consultar productos
Consultar actividades
Añadir actividades
Eliminar actividades
Modificar actividades
Enviar spam
Login
Logout
Usuario admin

4.5. Diagrama de actividad

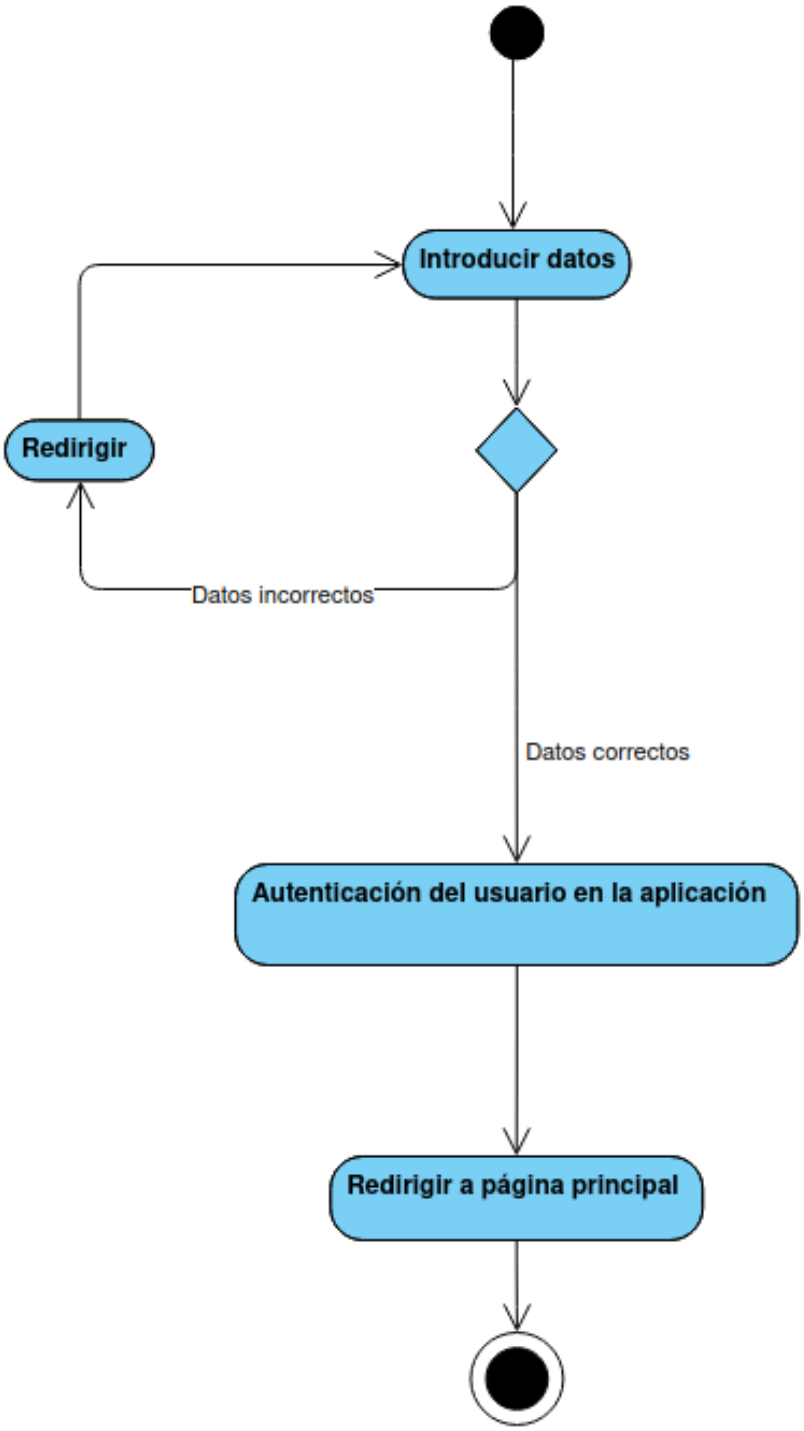
En este punto vamos a representar las funcionalidades más complejas de la aplicación, mediante el uso de diagramas de actividad.

4.5.1. Registrarse



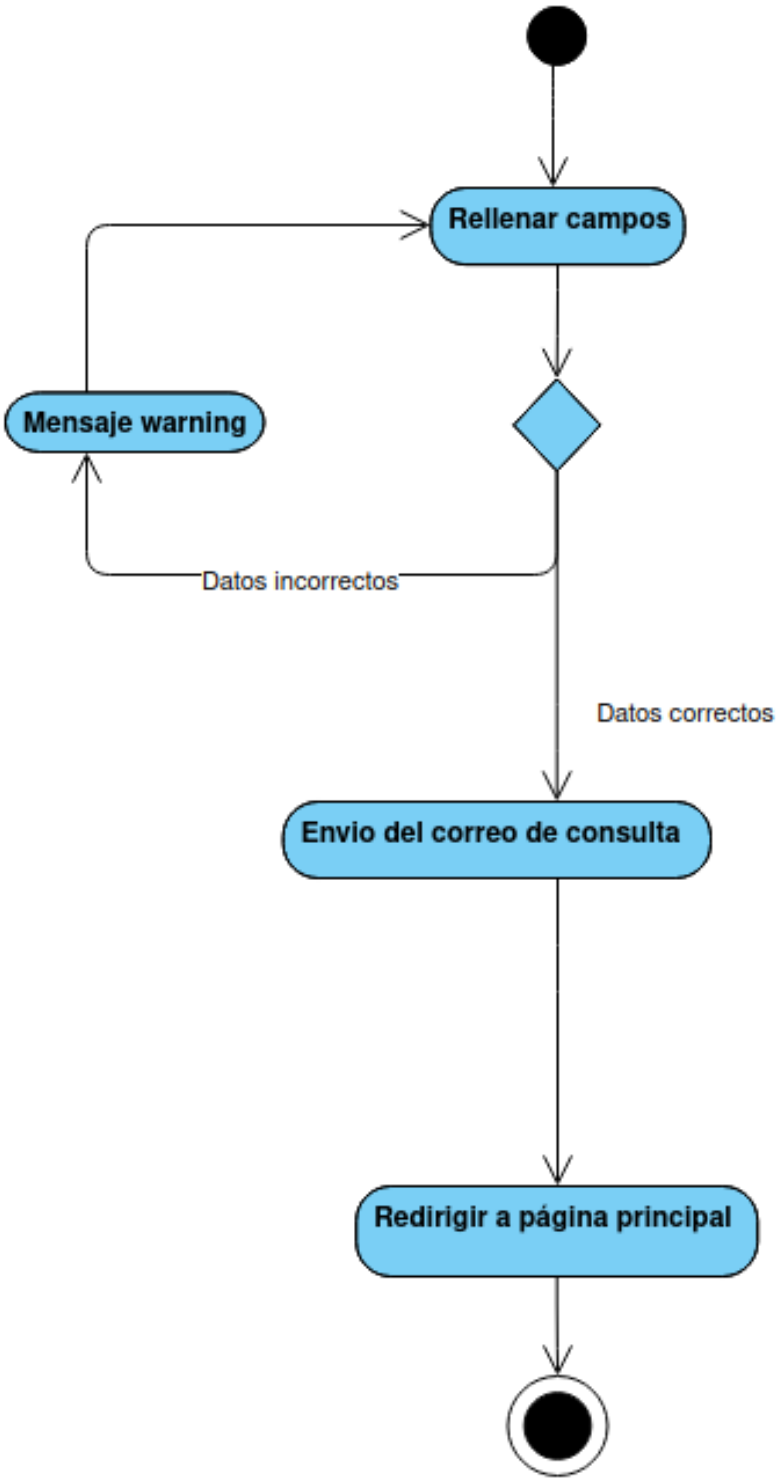
El usuario deberá introducir los datos para registrarse, si alguno de los datos introducidos es incorrecto se le redirigirá a la página de registro, si todos los datos son correctos la aplicación registrará al usuario en la base de datos, después le hará el login y será redirigido a la página principal de la aplicación.

4.5.2. Autenticarse



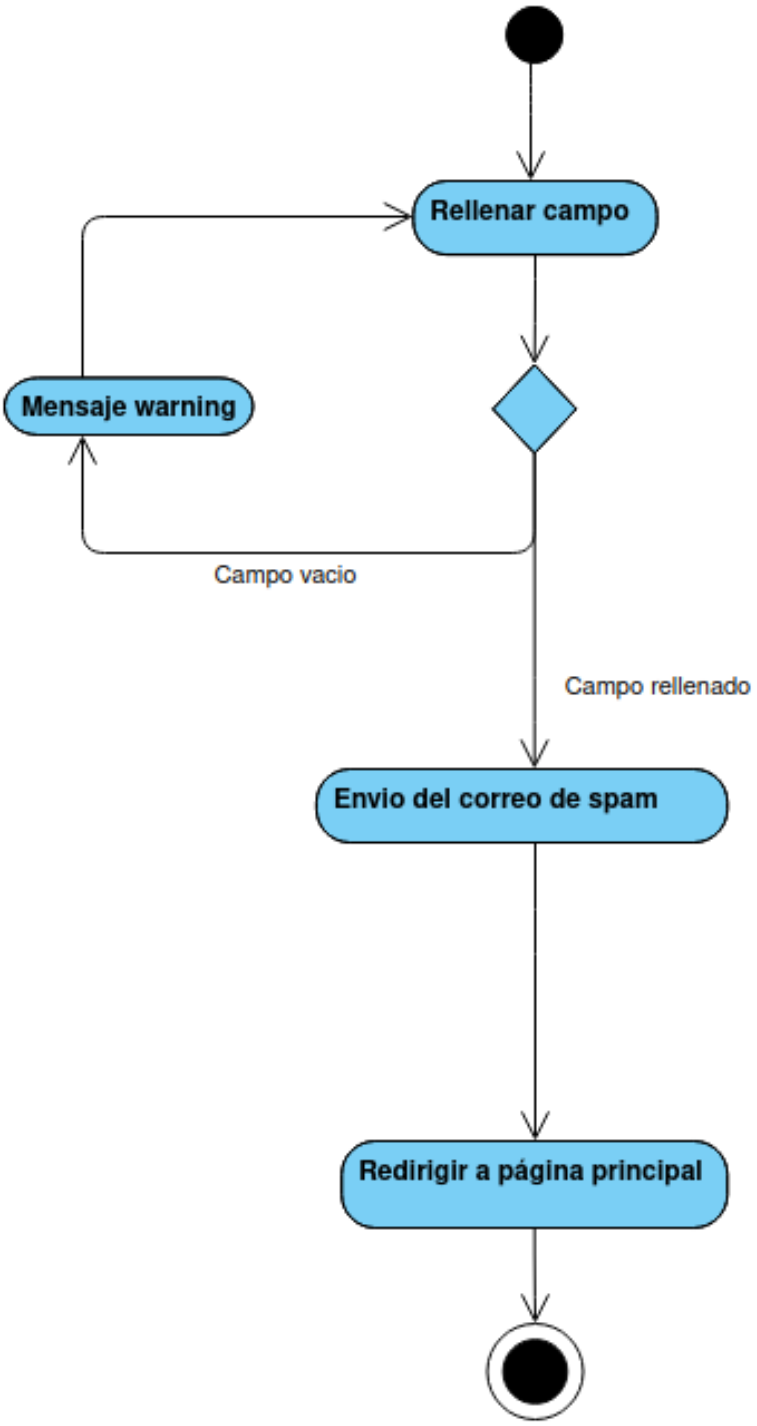
El usuario, que debe haberse registrado previamente en la aplicación, deberá introducir los datos para poder autenticarse en la aplicación, si alguno de los datos es incorrecto será redirigido a la página de login de nuevo, si los datos fueran correctos la aplicación procederá con la autenticación del usuario, para después redirigirlo a la página principal de la aplicación.

4.5.3. Enviar correo de consulta



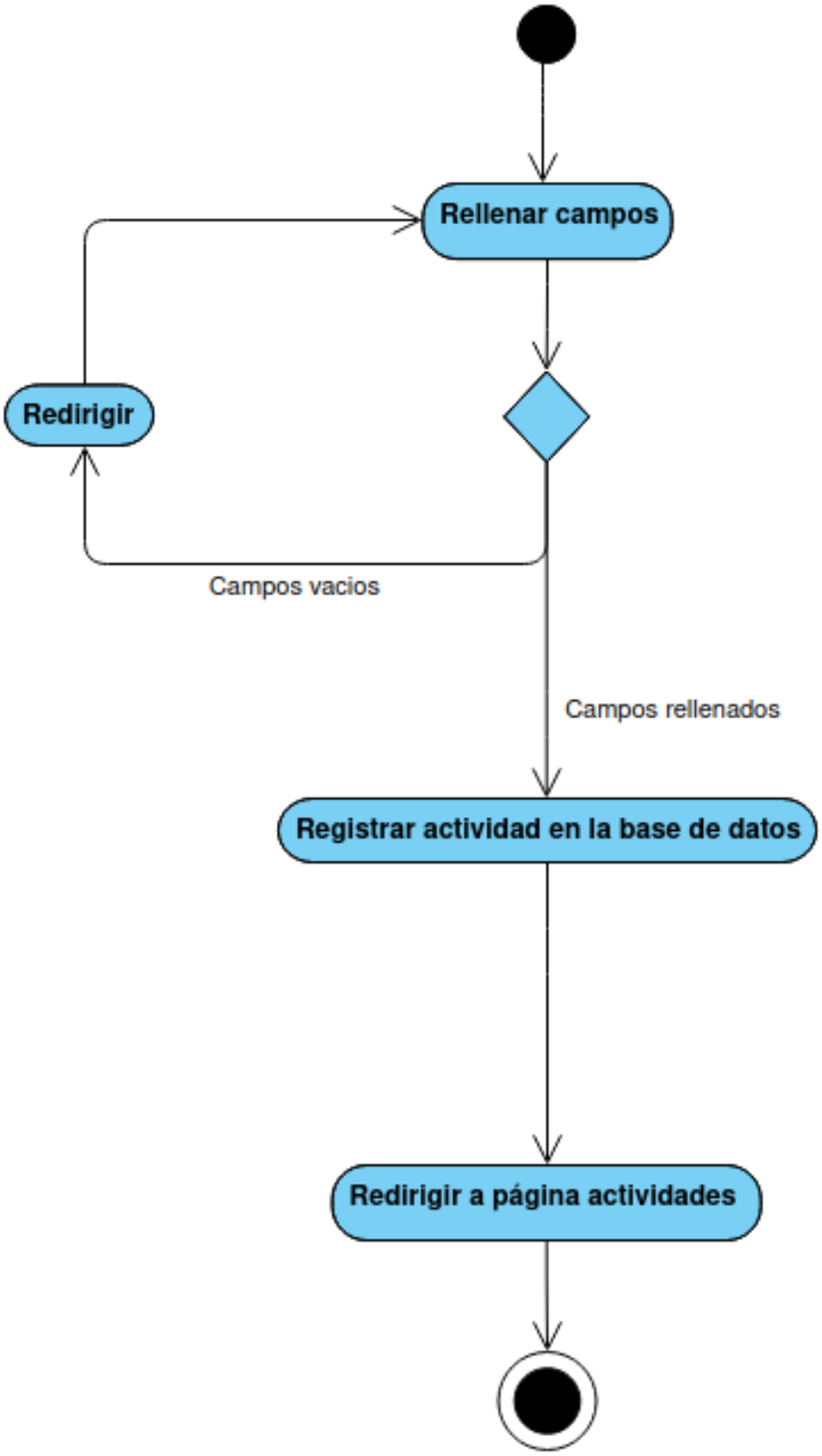
El usuario deberá rellenar los campos para enviar la consulta, si alguno de los campos es incorrecto o vacío, la aplicación mostrará un mensaje de alerta, diciendo el motivo del error por el cual no se ha enviado el correo, si todos los campos han sido rellenados correctamente, el correo será enviado y el usuario será redirigido a la página principal de la aplicación.

4.5.4. Enviar correo de spam



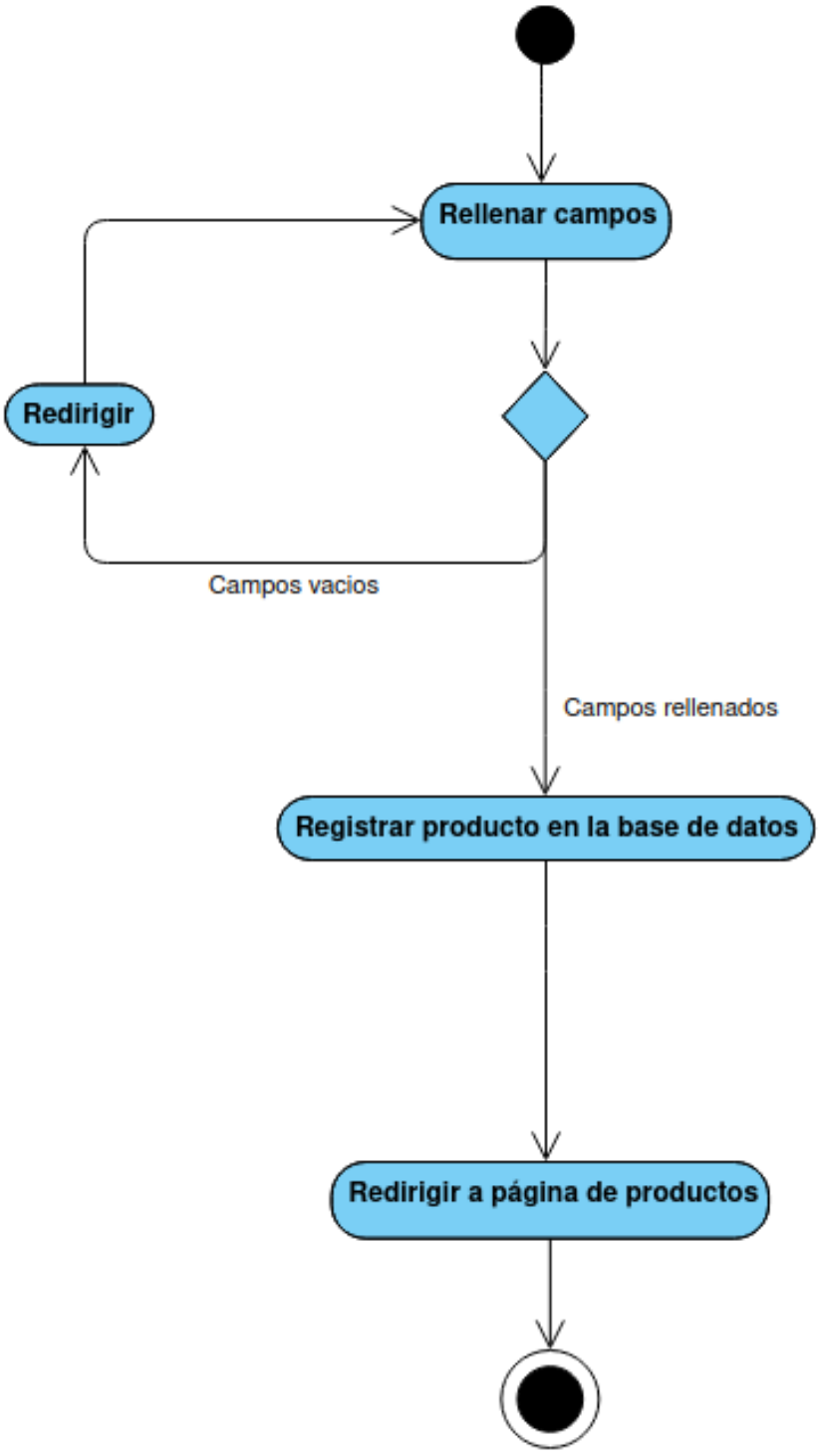
Los usuarios registrados que posean permisos de administrador, deberán rellenar el campo con el contenido del mensaje que será enviado a todos los usuarios registrados, si el campo está vacío, se le mostrará un mensaje indicando que no se ha podido enviar el correo de spam, ya que el campo de contenido no ha sido rellenado, si el campo ha sido rellenado correctamente se enviará el correo a todos los usuarios registrados, y el usuario con permisos de administrador será redirigido a la página principal.

4.5.5. Añadir actividad



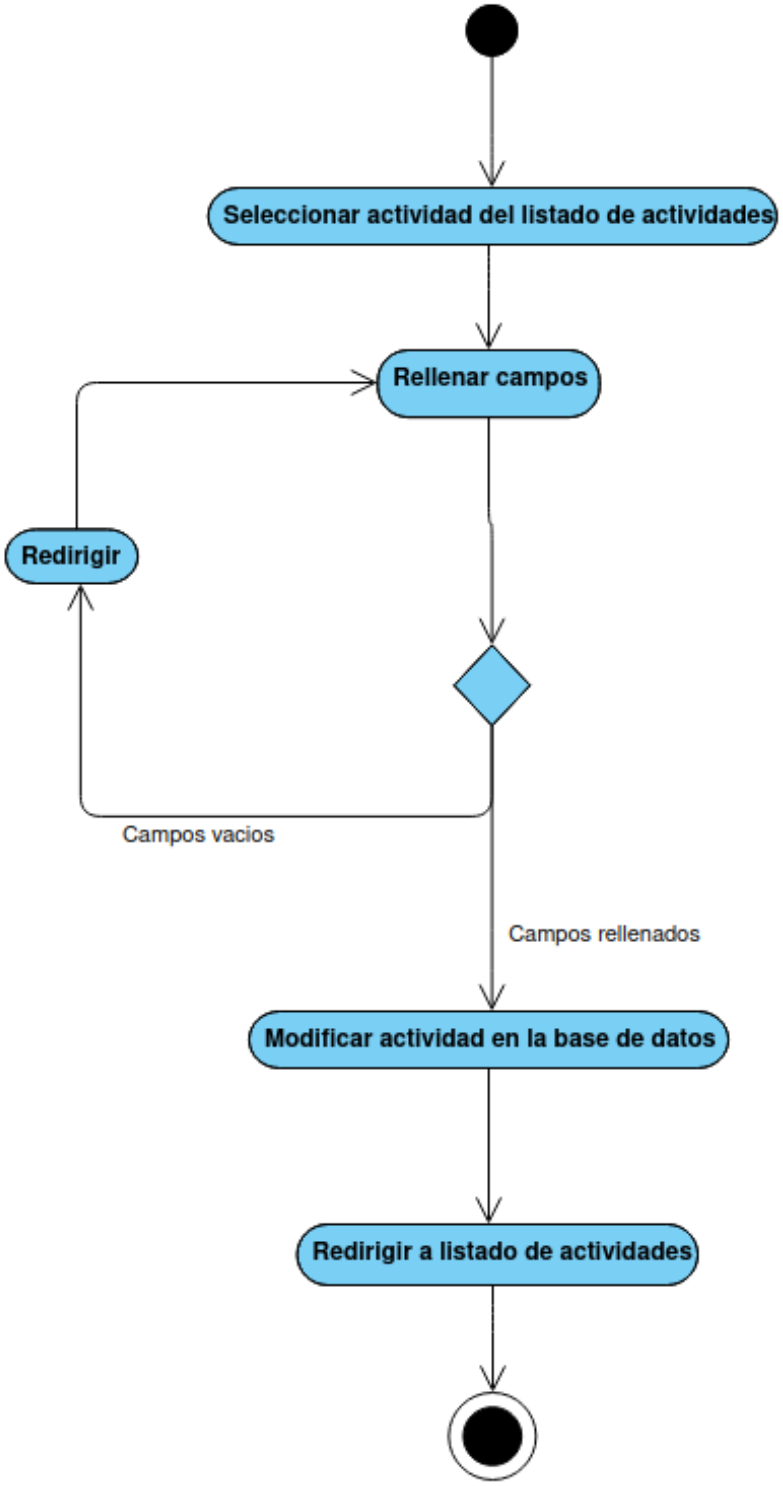
El usuario registrado con permisos de administrador tendrá que rellenar los campos de la actividad, si algún campo está vacío, será redirigido a la página de crear actividad, si todos los campos están rellenados, se creará la nueva actividad en la base de datos y el usuario será redirigido a la ventana de actividades.

4.5.6. Añadir producto



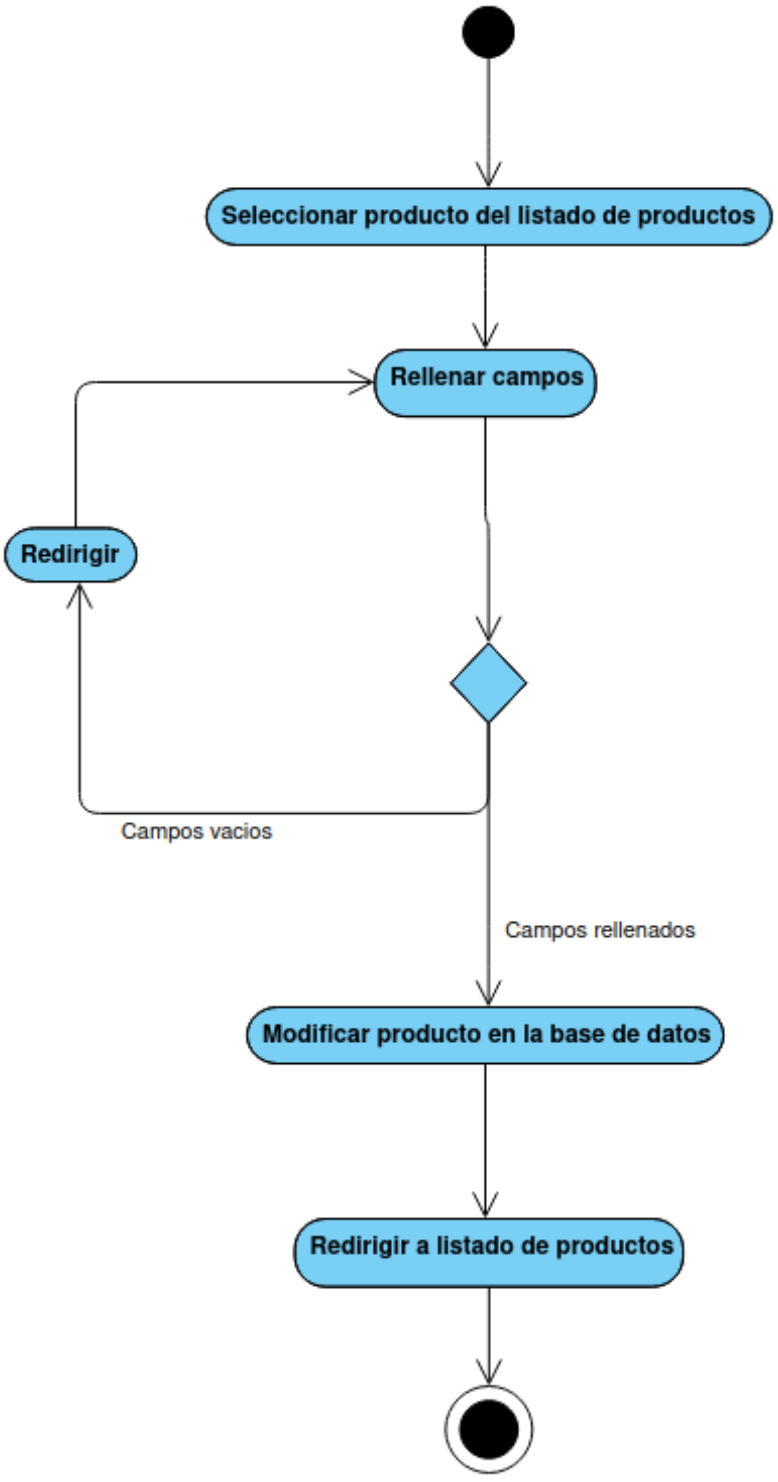
El usuario registrado con permisos de administrador tendrá que rellenar los campos del producto, si algún campo está vacío, será redirigido a la página de crear producto, si todos los campos están rellenados, se creará el nuevo producto en la base de datos y el usuario será redirigido a la ventana de productos.

4.5.7. Modificar actividad



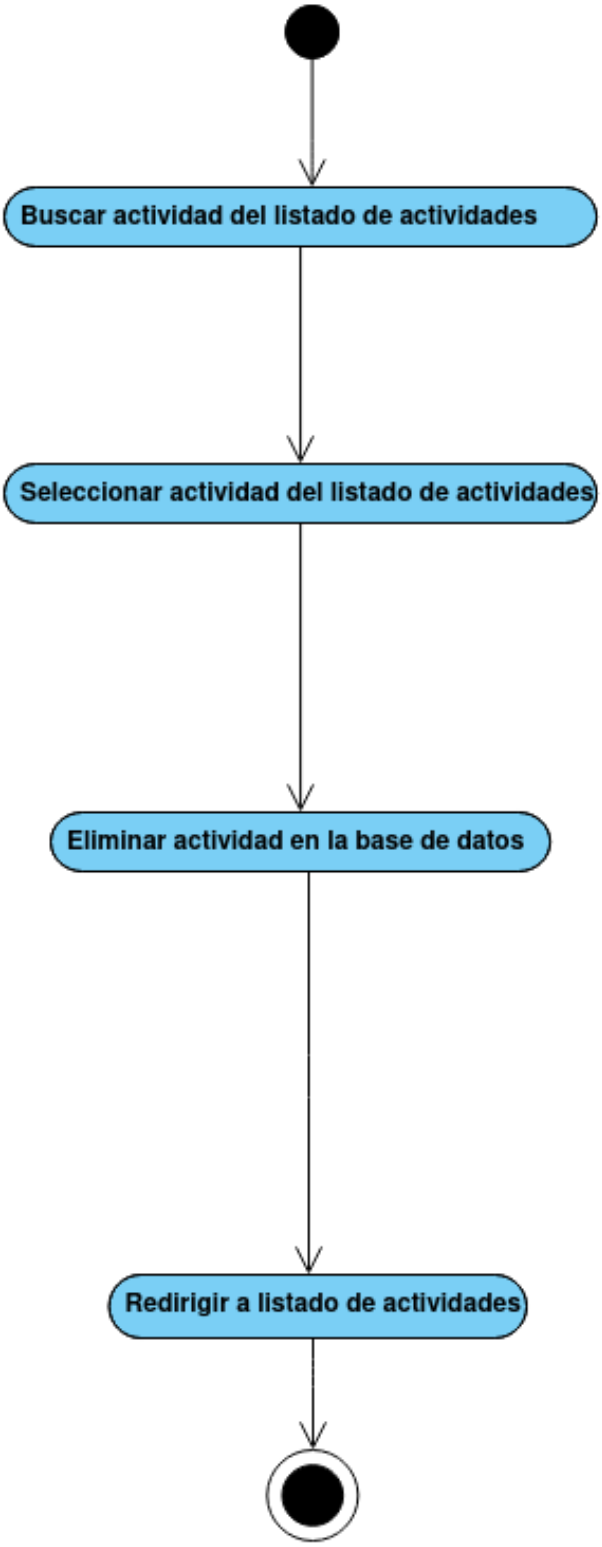
El usuario registrado con permisos de administrador tendrá que buscar la actividad en el listado, seleccionarla y rellenar los campos a modificar, si algún campo se deja vacío será redirigido a la página del listado de actividades, si los campos son rellenados correctamente, se modificará la actividad en la base de datos y el usuario será redirigido a la ventana del listado de actividades.

4.5.8. Modificar producto



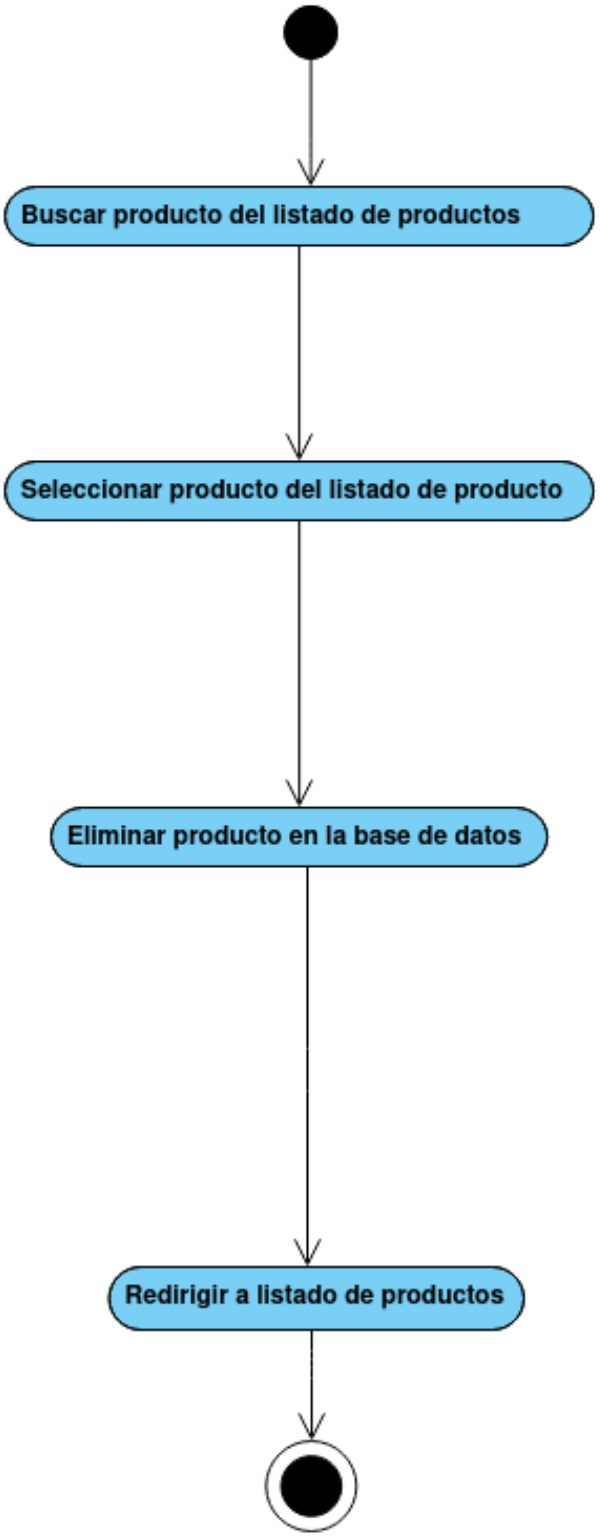
El usuario registrado con permisos de administrador tendrá que buscar el producto en el listado, seleccionarlo y rellenar los campos a modificar, si algún campo se deja vacío será redirigido a la página del listado de productos, si los campos son rellenados correctamente, se modificará el producto en la base de datos y el usuario será redirigido a la ventana del listado de productos.

4.5.9. Eliminar actividad



El usuario registrado con permisos de administrador tendrá que buscar la actividad a eliminar y seleccionarla, después se eliminará dicha actividad de la base de datos y el usuario será redirigido a la página del listado de actividades.

4.5.10. Eliminar producto



El usuario registrado con permisos de administrador tendrá que buscar el producto a eliminar y seleccionarlo, después se eliminará dicho producto de la base de datos y el usuario será redirigido a la página del listado de productos.

5. Diseño

5.1. Introducción

El patrón de diseño que se ha empleado en esta aplicación, es el que se conoce como patrón MVC o Modelo-Vista-Controlador. Dicho patrón de arquitectura de software, mantiene una separación entre la de la interfaz del usuario, la capa de la lógica de negocio y la capa de los datos de la aplicación. Mediante el uso de este patrón, se busca facilitar las tareas de desarrollo y mantenimiento, ya que tiene una clara separación entre los tres módulos mencionados anteriormente.

5.2. Patrón de diseño

5.2.1. Modelo

En el patrón de diseño MVC, la capa del Modelo es la encargada de controlar y manipular los datos y la lógica de negocios. Define los datos que la aplicación debe tener, y si se produce algún cambio en cualquiera de los datos, el modelo, generalmente, será el encargado de notificar a la capa de Vista o al Controlador, según sea necesario.

En el caso de nuestra aplicación, de esta capa se encargará el SGBD. Como ya se mencionó en un apartado anterior, Django usa por defecto un gestor (aunque puede usarse otro), que será el que realizará las modificación en la BBDD, mediante el uso de los modelos que se creen mediante código en la aplicación. A continuación se definirán las clases que se emplean en la aplicación:

• Usuario: Contiene los datos de los usuarios que realicen un registro en nuestra aplicación.

• Admin: Esta clase contiene los datos de los usuarios que tengan permisos de administrador, que serán los que tendrán acceso a algunas de las funcionalidades de la aplicación.

• Actividad: En esta clase se guardan los datos de las actividades ofertadas por el centro.

• Producto: En esta clase están guardados los datos de los productos puestos a la venta en la aplicación.

5.2.2. Vista

En el patrón MVC, la capa de Vista es en la que se define como deben mostrarse los datos en la interfaz de usuario de la aplicación. Más concretamente, es la parte de la aplicación que los usuarios podrán ver (la parte visual), donde se encuentran los componentes como: botones, cuadros de texto, labels, etc. En Django generalmente se utilizan plantillas de HTML, para definir el diseño que tendrán las aplicaciones web, también se puede hacer uso de bibliotecas como Bootstrap y archivos CSS para mejorar el aspecto visual de los componentes definidos en los documentos de HTML.

Una parte importante de la capa de Vista en cualquier tipo de aplicación web, es que el diseño de la misma tiene que ser responsive, es decir, que la aplicación web sea capaz de redimensionarse y redistribuir los componente, de forma que se adapten a las dimensiones del dispositivo en el que se utilice la aplicación, permitiendo al usuario una mejor y más cómoda experiencia. De esta manera la aplicación deberá poder ser visualizada en una pantalla de ordenador, una tablet o un smartphone.

5.2.3. Controlador

En el patrón MVC, la capa de Controlador posee una parte de la lógica, que es la encargada de actualizar tanto la capa de Modelo, como la capa de la Vista, o las dos a la vez; como respuesta a las entradas que los usuarios de la aplicación puedan realizar mediante el uso de la misma. De manera que gestiona los flujos de datos y las transformaciones para adaptar los datos, en función de las necesidades que tengan las otras dos capas.

En Django la capa de Controlador se encuentra en los documentos de rutas y de gestión de entradas de los usuarios. Es decir, en cualquier aplicación desarrollada en Django tenemos un o unos archivos urls.py y un o unos archivos views.py.

El archivo urls.py será el encargado de gestionar las rutas de la aplicación, o sea el enrutamiento. En el caso de nuestra aplicación, que contiene varias aplicaciones y por lo tanto varios archivos urls.py, los urls.py deberán estar incluidos en el urls.py de la aplicación principal.

Los archivos views.py, a pesar de su nombre, no se encargan de las Vistas, son los encargados de gestionar las entradas (request) que produzcan los usuarios y crear la respuesta que haya sido diseñada por los desarrolladores en los archivos views.py, como es lógico, desde los archivos views.py los desarrolladores pueden interactuar tanto con elementos de las BBDD, como con elementos que se encuentren en la interfaz de usuario, es decir, con elementos de las capas de Vista y Modelo, de esta manera les permite gestionar y permitir la interacción entre los usuarios y la aplicación, proveyendo a la aplicación de respuestas frente a cualquier tipo de acción (entrada) que realicen los usuarios.

La combinación de los archivos views.py y urls.py, es lo que crea la capa de Controlador en una aplicación desarrollada en Django, ya que también existirán respuestas que consistan en redirigir a una página distinta, y en esas situaciones es donde el archivo urls.py, archivo que gestiona todas las urls de la aplicación, será necesario. De esta manera, estos dos archivos tendrán control absoluto sobre la gestión de las interacciones entre los usuarios y la aplicación.

6. Implementación

6.1. Entorno de desarrollo

Para empezar este proyecto, el primer paso fue tomar la decisión de cambiar el sistema operativo del ordenador portátil en el que se iba a desarrollar la aplicación. Esta motivación vino dada por el hecho de que para ejecutar y crear un programa en Django, se tienen que usar instrucciones por línea de comando. Como en la universidad utilizamos línea de comandos en escritorios remotos Linux, ya se partía de una base de conocimientos de la misma, mientras que en Windows no existían esos conocimientos por la falta de necesidad, porque Windows facilita su uso sin tener que usar esta aplicación. Por no hablar de que, por el hecho mencionado anteriormente, la potencia de la línea de comandos de Linux es superior a la de Windows.

Por eso mismo, el primer paso para empezar el proyecto fue crear un USB booteable, es decir, un USB que permite al ordenador arrancar desde el mismo, permitiendo utilizar el sistema operativo cuya imagen de arranque se encuentra en dicho USB, y permitiendo también la instalación de dicho sistema operativo en nuestro ordenador.

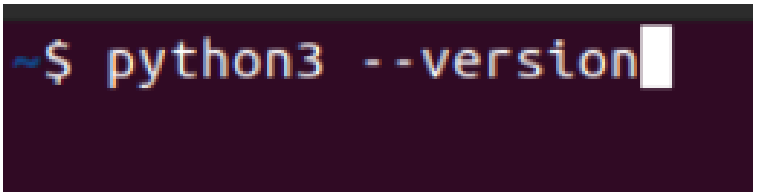
En el caso de este proyecto, se utilizó el programa Rufus para la creación del USB booteable, ya que es uno de los más completos y sencillos de usar, entre los programas para la creación de este tipo de USBs. Luego simplemente había que disponer de un USB con capacidad suficiente de almacenamiento y la imagen del sistema operativo que se desee instalar, en este caso Linux y más concretamente Ubuntu en la versión 20.10, y con las últimas actualizaciones de seguridad para este sistema operativo.

También será necesario instalar el lenguaje de programación Python[5] para poder utilizar el entorno de desarrollo. Para poder instalar el lenguaje de programación habrá que ejecutar las siguientes instrucciones:

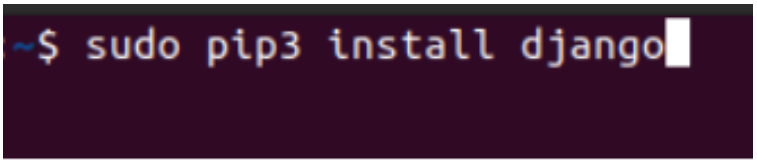
```
sudo apt-get update

sudo apt-get install python3.6
```

Como nuestra versión de ubuntu es una de las más actuales, se podrá instalar la versión de Python 3 que aparece en la imagen, habiendo previamente ejecutado el comando update para que se lleven a cabo todas las actualizaciones pendientes.

En el caso de que queramos comprobar la versión de Python 3 que tenemos instalada en el ordenador, sólo tendremos que ejecutar el siguiente comando:

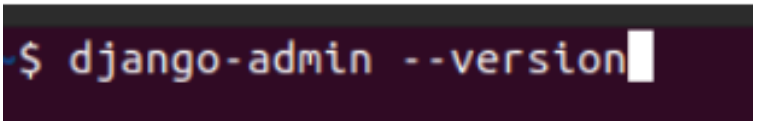


Por último, también será necesario instalar el entorno de desarrollo, en este caso Django, con el que queremos trabajar. Para ello tendremos que decidir primero si instalarlo con Python 2 o Python 3, en el caso de este proyecto se instaló el entorno con Python 3. Para ello se tiene que emplear la siguiente línea de comando:



En el caso de que quisiéramos instalar Django con Python 2 en lugar de Python 3, simplemente tendríamos que usar la misma instrucción, pero cambiando el pip3 que aparece en la imagen por pip2.

Si queremos comprobar la versión de Django que hemos instalado, tendremos que ejecutar la siguiente instrucción:

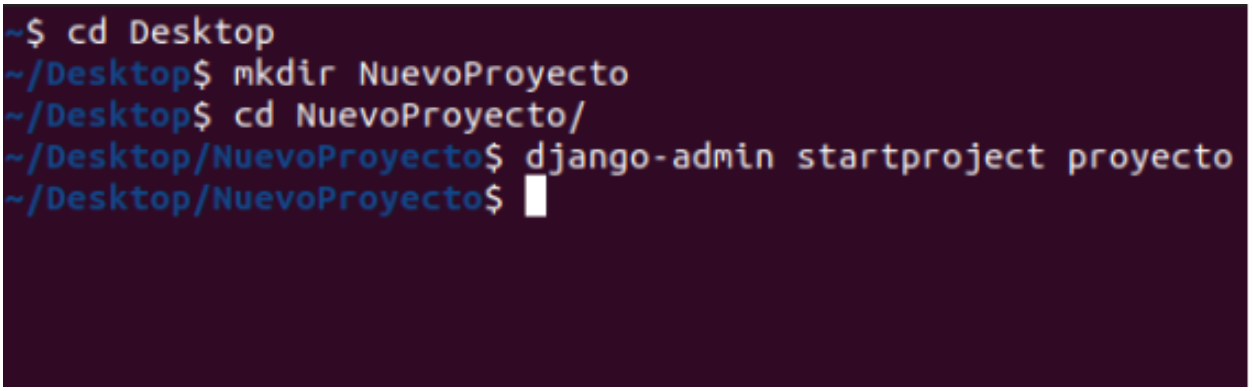


Con esto ya tendríamos instalado el entorno y el lenguaje de programación, en la versión que más nos interese utilizar.

6.2. Creación y configuración del proyecto

Una vez instalado correctamente el sistema operativo Linux en el ordenador portátil, solo había que empezar a crear el proyecto. Django facilita bastante la creación de proyectos, ya que no hay que configurar nada previamente para poder empezar a utilizarlo, en el mayor de los casos, esos cambios en la configuración deberán realizarse conforme avanza el desarrollo de la aplicación.

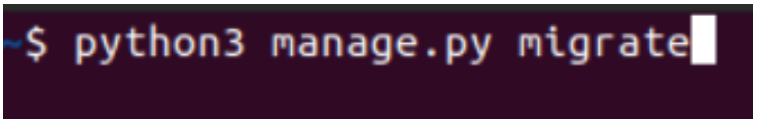
Para la creación de un proyecto en Django habrá que utilizar las siguientes instrucciones en la línea de comando:



Tal y como aparece en la imagen, habrá que especificar la ubicación en la que queramos crear el directorio. El comando *cd* se utiliza para navegar entre carpetas, con esta instrucción podemos buscar la carpeta en la que queramos ubicar el proyecto. El comando *mkdir* se utiliza para crear una carpeta, esto puede resultar muy útil si nuestra intención es crear una nueva carpeta, que será exclusivamente para almacenar el proyecto.

Una vez hayamos decidido la ubicación exacta del proyecto, tendremos que ejecutar la instrucción *django-admin startproject* seguida del nombre que le queramos poner al proyecto. Una vez ejecutada la instrucción, Django creará un proyecto el cual tendrá como nombre de la carpeta principal, el que hemos utilizado en la línea de comando para crear el propio proyecto. A partir de este punto, cualquier instrucción que tenga que afectar a nuestro proyecto, deberemos realizarla desde la carpeta del proyecto que acabamos de crear por comando.

Una vez tengamos el proyecto creado, tendremos que iniciar la base de datos. Para lo cual tendremos que ejecutar la siguiente línea:



Cuando se ejecuta esta línea por primera vez, se produce la creación y arranque de la base de datos, y se realizan todas las migraciones principales que debe hacer, para el correcto funcionamiento de la misma.

Y con esto ya podríamos empezar a desarrollar nuestra aplicación en Django, pero también existen otras instrucciones que eventualmente tendremos que emplear. Como pueden ser las siguientes:

Con la instrucción *createsuperuser* podremos crear un usuario con permisos de administrador, con el que podremos acceder a la página de administrador que tienen, desde que son creadas, todas las aplicaciones desarrolladas en Django. Desde la página de administrador, podremos crear otros usuarios o cualquier tipo de objeto que haya sido registrado en un modelo de base de datos desde nuestra aplicación.

Esto podría resultar muy útil, ya que también nos permitirá modificar o eliminar cualquiera de estos objetos, así como realizar búsquedas para encontrar el objeto que estamos buscando. La diferencia entre un usuario y un superusuario es que a este último se le marca con la etiqueta staff, lo que también nos permite diferenciar, una vez realizado un posible login en nuestra aplicación, a unos usuarios que tengan permisos de administrador de otros que no tengan esos permisos.

```
$ python3 manage.py makemigrations
```

La instrucción *makemigrations* tiene mucha importancia cuando ya hemos creado nuestros modelos de bases de datos en la aplicación. Esta instrucción se utiliza para que nuestra aplicación realice las migraciones de forma local, con la intención de luego poder registrarlas con el comando *migrate* en la aplicación, de esta manera se podrían visualizar estos cambios cuando se ejecute la aplicación.

```
$ python3 manage.py runserver
```

La instrucción *runserver* es una de las más importantes y de las que más se utilizan cuando se desarrolla una aplicación en Django. Ya que es la que arranca el servidor en el que se encuentra la aplicación.

Cuando ejecutamos esta instrucción, la aplicación de línea de comandos nos muestra el enlace del servidor en el que se está ejecutando la aplicación, permitiendo que podamos abrir el enlace desde la propia ventana de línea de comandos y también nos muestra el comando que tendremos que emplear si queremos dejar de ejecutar el servidor.

En el caso de que hubiera algún error de compilación o ejecución este se nos mostrará también por la línea de comandos.

```
$ python3 manage.py help
```

En el caso de que tengamos dudas o no nos acordemos de las instrucciones que Django posee para ejecutarlas desde la línea de comandos, podemos usar el comando *help*, que al ejecutarlo mostrará todas las posibles instrucciones que podemos ejecutar, con respecto a Django.

Y si quisiéramos saber la descripción de una instrucción en concreto, solo tendríamos que añadir el nombre de la instrucción a continuación de *help*, por ejemplo, si queremos saber lo que hace el comando *makemigrations*, tendremos que escribir la siguiente instrucción:

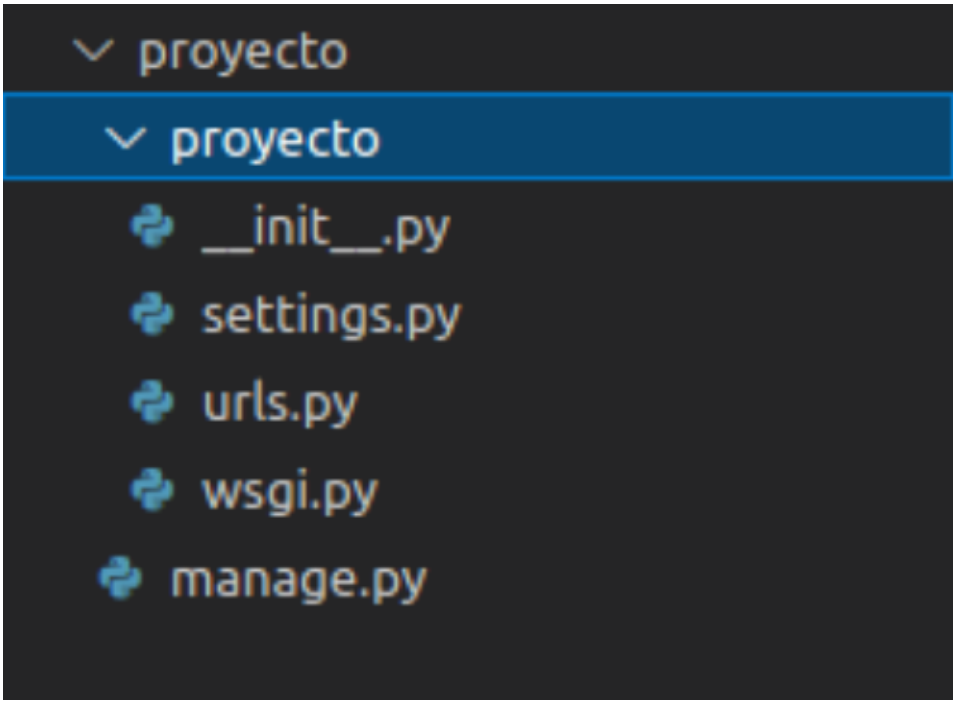
```
usage: manage.py makemigrations [-h] [--dry-run] [--merge] [--empty]
                                [--noinput] [-n NAME] [--no-header] [--check]
                                [--version] [-v {0,1,2,3}]
                                [--settings SETTINGS]
                                [--pythonpath PYTHONPATH] [--traceback]
                                [--no-color] [--force-color]
                                [app_label [app_label ...]]

Creates new migration(s) for apps.
```

Y tal y como aparece en la imagen, nos describe lo que hace el comando y nos dice las posibles variantes que tiene el comando.

6.3. Estructura de un proyecto Django

La estructura de una aplicación web desarrollada en Django puede ser muy diversa, ya que el desarrollador puede crear varias carpetas, ya sea con la intención de organizar el proyecto dividiéndolo en partes, creando carpetas específicas para los templates o carpetas para ubicar las imágenes o archivos que sirvan para ayudar en el diseño, como un archivo CSS.

Pero la estructura básica de un proyecto desarrollado en Django, sería el que se muestra en la siguiente imagen.



Dentro de la subcarpeta llamada proyecto y dentro de la carpeta general de proyecto tenemos cinco archivos que sirven para las siguientes funciones:

• El archivo __init__.py sirve para que Python identifique a la carpeta contenedora de este archivo como un directorio de paquetes para el propio Python, o sea, que indica que el paquete contiene un módulo, que debe ser tratado como tal, permitiendo importar archivos como parte del módulo.

• El archivo settings.py, tal y como indica su nombre, es un archivo de configuración, que sirve para especificar si activar el depurador que lleva Django incorporado, la lista con las aplicaciones que va a ejecutar nuestro proyecto (en el caso de que tenga más de una), cual de las distintas bases de datos aceptadas por Django vamos a utilizar, el idioma en el que se ejecutara la aplicación, la URL de los archivos estáticos o multimedia que vaya a utilizar nuestra aplicación, entre otras cosas.

• El archivo urls.py sirve para registrar las URLs que vaya a utilizar nuestra aplicación, asociando el método, con su respectiva entrada, el nombre que queramos darle, entre otras cosas. Cualquier URL que vaya a ser utilizada por nuestra aplicación, deberá ser registrada previamente en este archivo, de lo contrario, al no encontrar la aplicación la URL no podrá acceder a ella y nos aparecerá el respectivo error de que la URL buscada no existe en la aplicación.

• El archivo wsgi.py, cuyas siglas significan Web Server Gateway Interface, sirve para especificar cómo se comunica un servidor web con una aplicación web, y también cómo pueden llegar a encadenarse distintas aplicaciones web para solucionar una determinada petición.

• El archivo manage.py sirve para ayudar con la administración del sitio web, gracias a este archivo podremos arrancar un servidor web desde nuestro ordenador, sin necesidad de tener que realizar ningún otro tipo de instalación.

Así como los archivos que hemos mencionado anteriormente son los creados automáticamente por Django, siempre que se crea un nuevo proyecto, tal y como hemos mencionado previamente, esta estructura no es la única que tienen las aplicaciones desarrolladas en este entorno de desarrollo.

En el caso de que quisiéramos crear un modelo de base de datos para la aplicación que queramos desarrollar, tendremos que crear un archivo dentro de la subcarpeta proyecto, llamado models.py. En este archivo tendremos que programar un modelo de base de datos como el siguiente.

```python
from django.db import models

class Tipo(models.Model):
    nombre=models.CharField(max_length=50)
    created=models.DateTimeField(auto_now_add=True)
    updated=models.DateTimeField(auto_now_add=True)

    class Meta:
        verbose_name="TipoProducto"
        verbose_name_plural="TiposProductos"

    def __str__(self):
        return self.nombre

class Producto(models.Model):
    nombre=models.CharField(max_length=50)
    tipo=models.ForeignKey(Tipo, on_delete=models.CASCADE)
    imagen=models.ImageField(upload_to="tienda", null=True, blank=True)
    precio=models.FloatField()
    stock=models.BooleanField(default=True)
    created=models.DateTimeField(auto_now_add=True)
    updated=models.DateTimeField(auto_now_add=True)

    class Meta:
        verbose_name="Producto"
        verbose_name_plural="Productos"
```

En este archivo podremos poner los campos que queremos que tenga nuestro modelo, el nombre que le queremos asignar, el tipo de cada uno de sus atributos, y valores como si queremos que el campo pueda ser vacío o que tenga valor nulo. Y una vez tengamos el modelo, ejecutando los comandos *makemigrations* y *migrate*, se habrá registrado nuestro modelo en la base de datos que estemos utilizando.

Si quisiéramos crear un formulario para nuestra aplicación, tendríamos que crear un archivo llamado forms.py para programar en ese archivo el o los formularios que queramos utilizar en nuestra aplicación. Un ejemplo sencillo de un formulario para una aplicación en Django sería el siguiente.

```python
from django import forms

class Contacto(forms.Form):
    nombre=forms.CharField(label="Nombre", required=True)
    email= forms.EmailField(label="Email", required=True)
    contenido=forms.CharField(label="Contenido",
    required=True, widget=forms.Textarea)
```

Es este archivo podremos especificar el tipo de formulario que queremos, dándole un nombre y asignando los campos que tendrá el formulario,el tipo del valor que introducir en el campo, el label que tendrá cada campo, si queremos que el campo pueda estar vacío o no, entre otros atributos que también se pueden especificar.

Y como último ejemplo, los templates que usar para la capa de Vista de nuestra aplicación, pueden ser creados en la misma subcarpeta que los modelos y los formularios, pero uno de los convencionalismos de Django es tener esos archivos en una carpeta llamada templates, la cual puede tener a su vez otras carpetas anidadas.

Por estos motivos la estructura de una aplicación Django es muy diversa, porque también podríamos tener más de una aplicación en nuestro proyecto y cada aplicación tendría las mismas complejidades que hemos mencionado anteriormente, así que la diversidad de la estructura se ampliará al número de aplicaciones que tenga la aplicación principal.

6.4. Panel de administración

En este apartado vamos a hablar de la importancia del panel de administración de Django. Como hemos mencionado en puntos anteriores, Django provee a los desarrolladores de un panel de administración, al cual podrán acceder después de haber creado un superusuario, es decir, después de haber ejecutado el comando createsuperuser.

Desde este panel los desarrolladores podrán administrar los objetos que hayan creado en los modelos de bases de datos, así como también crear usuarios.

Este apartado puede ser muy útil en el caso de que la aplicación no permita gestionar los objetos de la base de datos, por el hecho de que son objetos que no deben ser gestionados, por lo que este panel de administración nos permitiría crear, modificar y eliminar dichos objetos, para posteriormente lanzar la aplicación con los objetos deseados por el cliente.

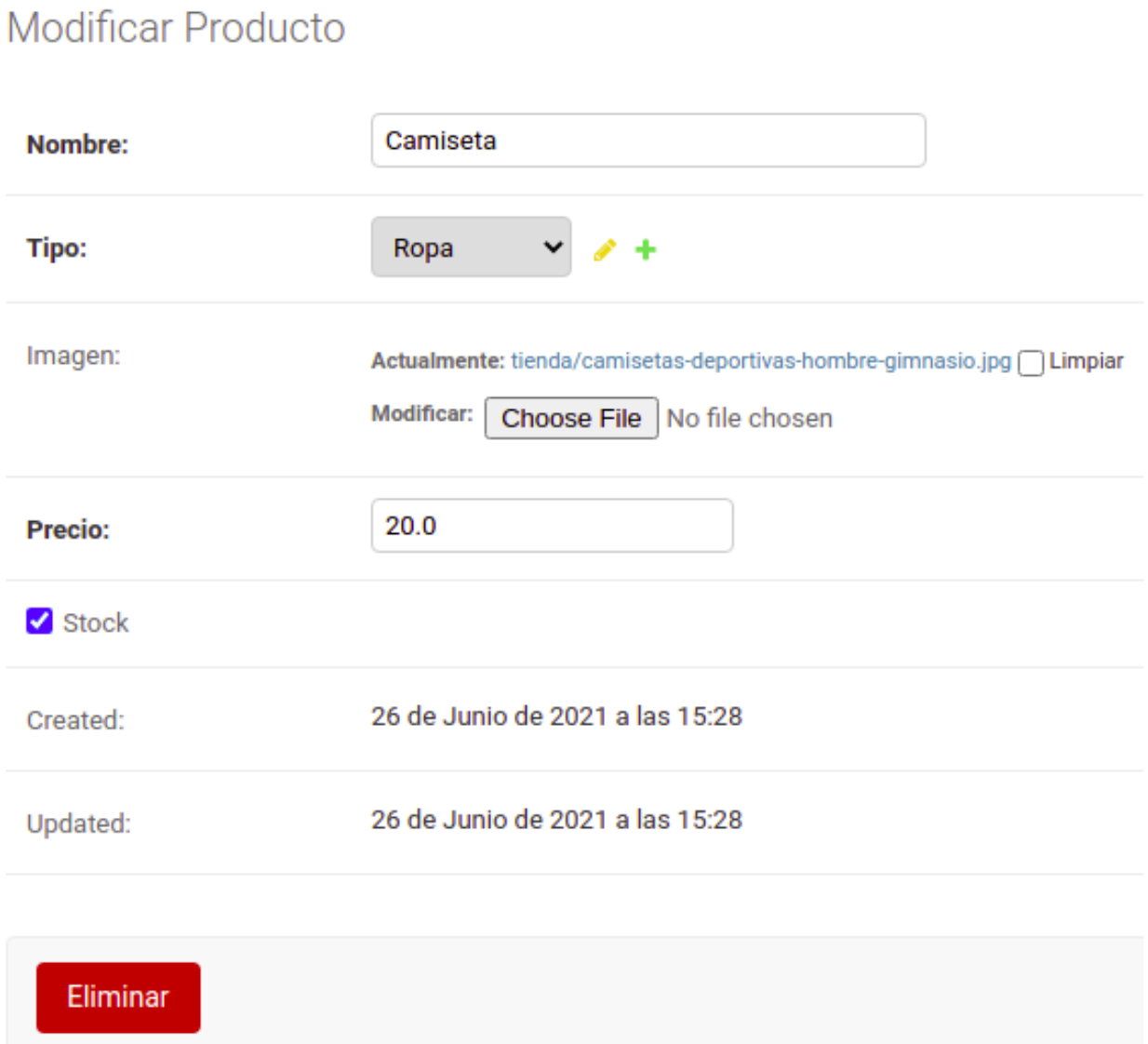
Un ejemplo de panel de administración podría ser el que se muestra en la siguiente imagen.

Tal y como se muestra en la imagen, podemos observar que nos permite añadir y modificar desde la ventana principal del panel de administración, tanto los usuarios y los grupos (en caso de que los haya), y los modelos creados para la aplicación, así como también nos permite observar cuales han sido las últimas acciones realizadas como administrador.

En el caso de que quisiéramos buscar un objeto concreto de nuestro modelo nos aparecerá una ventana con un listado de los objetos, un botón con nos permitirá añadir un nuevo objeto al modelo, eliminar los objetos que hayamos seleccionado y si accedemos a uno de ellos podremos modificarlo.

La ventana que nos aparecería si accediéramos a un modelo sería la que aparece en la siguiente imagen.

En el caso de que accediéramos a uno de los objetos de la base de datos, veríamos la siguiente ventana.

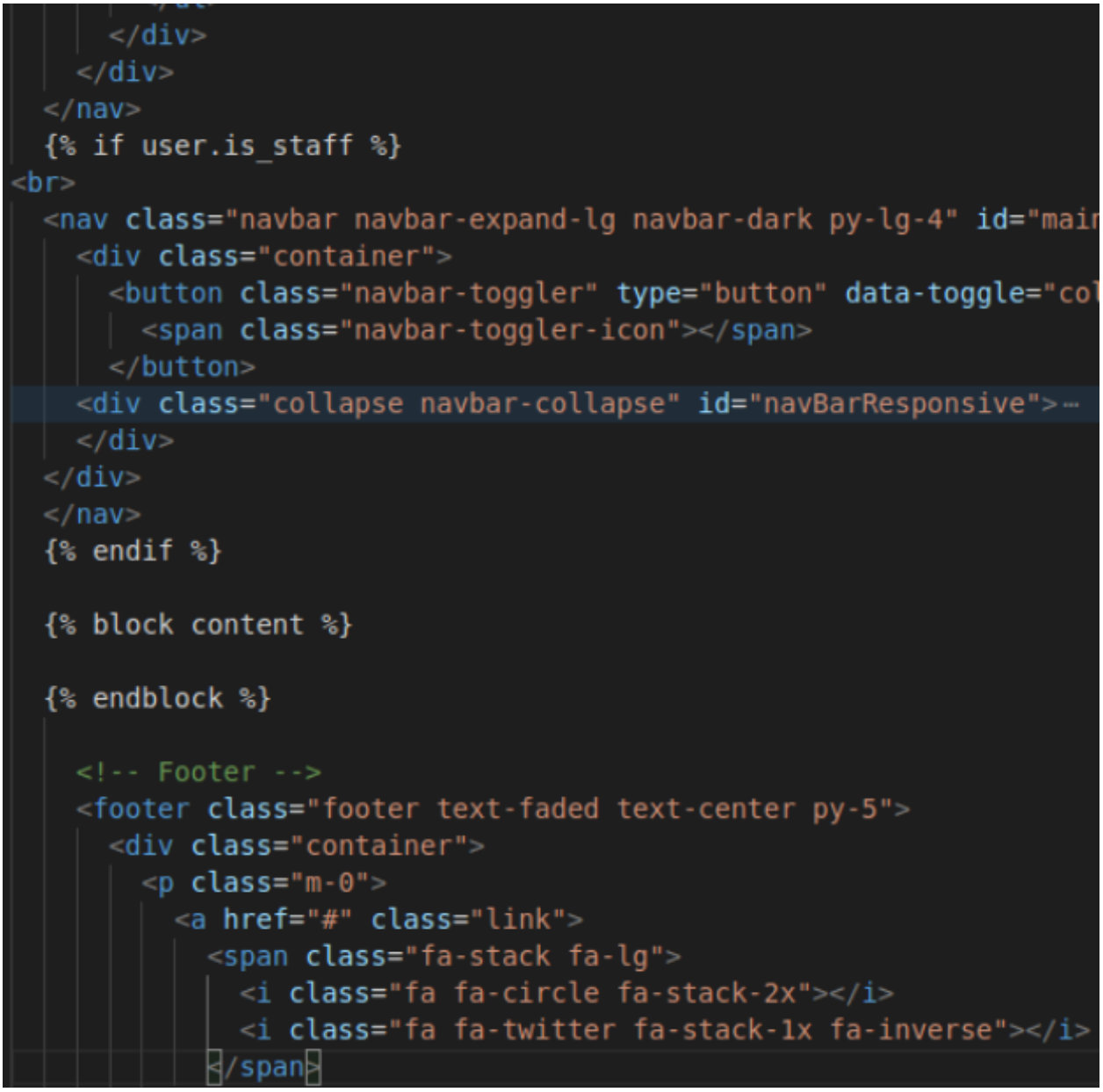


6.5. Templates

Tal y como hemos mencionado anteriormente, para la parte de diseño de aplicaciones web, Django utiliza unos archivos llamados templates (plantillas). Los templates son archivos de texto que se utilizan para definir la estructura y el diseño de las aplicaciones creadas en Django, que funcionan igual que una página programada en HTML.

Los templates[6] en Django pueden heredar de otro template, al cual se le denomina "base", esto se utiliza para que una misma aplicación web tenga un diseño homogéneo y similar en cada una de sus páginas, ofreciendo a los usuarios de la aplicación un diseño más atractivo visualmente. Para la herencia de plantillas, entre otras funcionalidades, Django utiliza las llamadas etiquetas de plantillas.

Un ejemplo de plantilla base sería el siguiente.



En la imagen anterior podemos observar cuatro etiquetas:

• La primera etiqueta es un {% if %}, la cual se utiliza para indicar a Django que todo lo que se encuentre entre esta etiqueta y la etiqueta del final de condicional sólo deberá aparecer u ocurrir en el caso de que se cumpla la condición que aparece en la etiqueta del if.

• La segunda etiqueta es el {% endif %} y sirve para, como se ha mencionado en el punto anterior, delimitar el tramo de código que entra en la instrucción condicional, pasada esta etiqueta la condición del if ya no tiene efecto.

• La tercera etiqueta es la de {% block content %}, se utiliza para marcar el inicio de la plantilla que va a heredar a la plantilla base, es decir, todo lo que aparezca por encima de esta etiqueta aparecerá en todas las plantillas que hereden de esta.

• La cuarta etiqueta es la de {% endblock %}, que indica el final de la plantilla que hereda a la base, o sea, a partir de esta etiqueta volverá a aparecer el contenido de la plantilla base en el resto de plantillas que hereden de esta. En resumen, cualquier plantilla que herede de la plantilla base aparecerá donde se encuentren las etiquetas de {% block content %} y {% endblock %}.

Además de eso, Django permite crear hojas de estilo CSS, para gestionar el diseño de los elementos HTML programados en los templates. Para hacer uso de los elementos creados en las hoja CSS, solo hay que utilizar la etiqueta "class" en el elemento en que se le quiera aplicar el estilo del CSS, con el identificador que se le haya puesto al estilo. Un ejemplo de esto sería lo que aparece en la siguiente imagen.

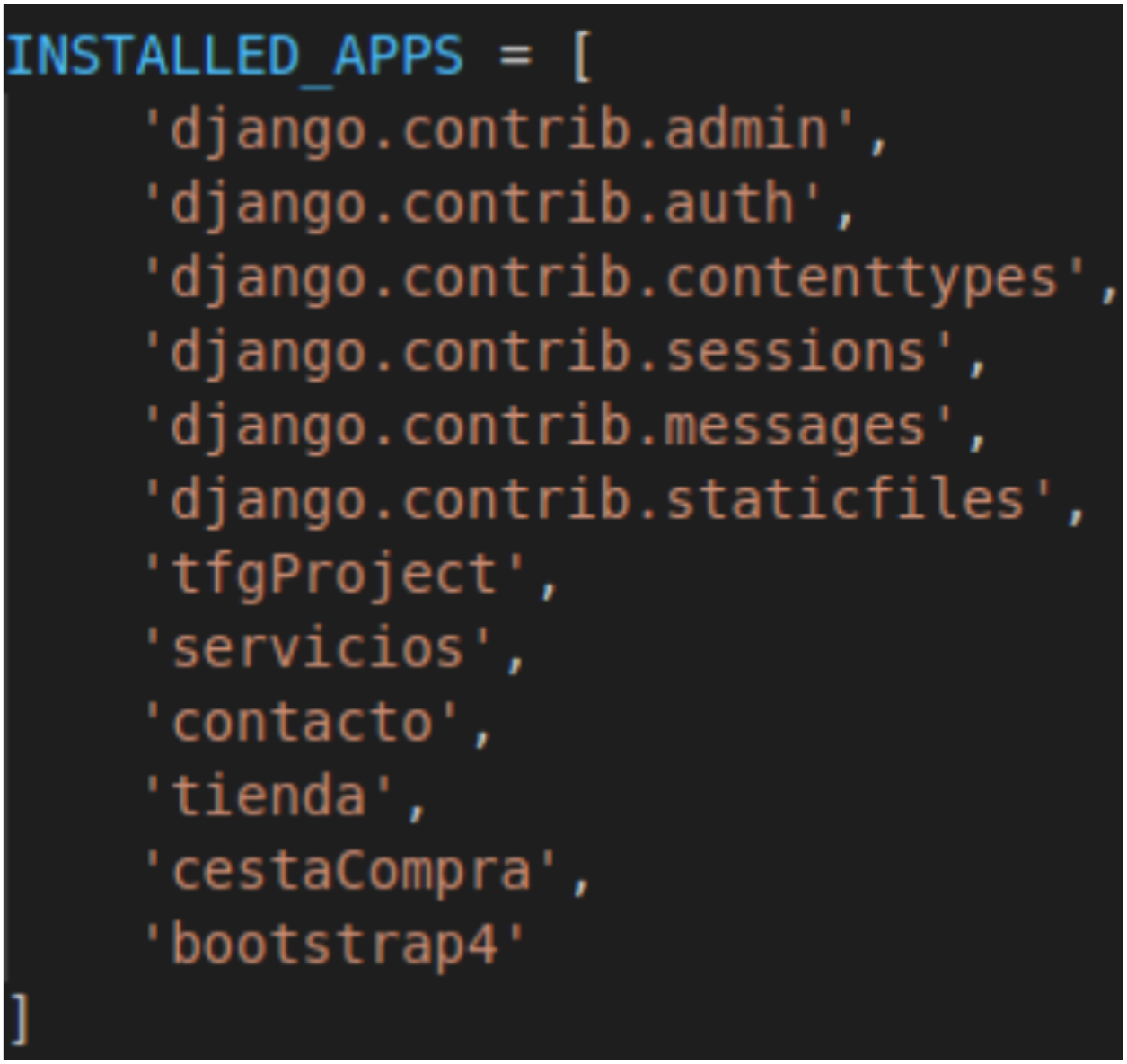
```
<div class="contactoForm">

    {% if "valido" in request.GET %}

    <p style="color: white;">Mensaje enviado correctamente.</p>

    {% endif %}
    {% if "novalido" in request.GET %}

    <p style="color: white;">Mensaje no enviado correctamente.</p>

    {% endif %}

    <form action="" method="POST" style="text-align: center;">
        {%csrf_token%}
        <table style="color: white; margin: 20px auto;">
            {{form.as_table}}
        </table>
        <input type="submit" value="Enviar" style="width: 150px;">
    </form>

</div>
```

Como se puede ver en la imagen, dentro de la etiqueta class podemos ver que se ha utilizado el estilo, que en el archivo CSS se le ha dado el nombre de "contactoForm".

A Django también se le puede añadir la biblioteca Bootstrap, para ello solo tendríamos que ejecutar la siguiente instrucción por línea de comando para instalarlo.

```
$ pip install django-bootstrap4
```

Y en la lista de INSTALLED_APPS de nuestro archivo settings.py añadir 'bootstrap4', como aparece en la siguiente imagen.



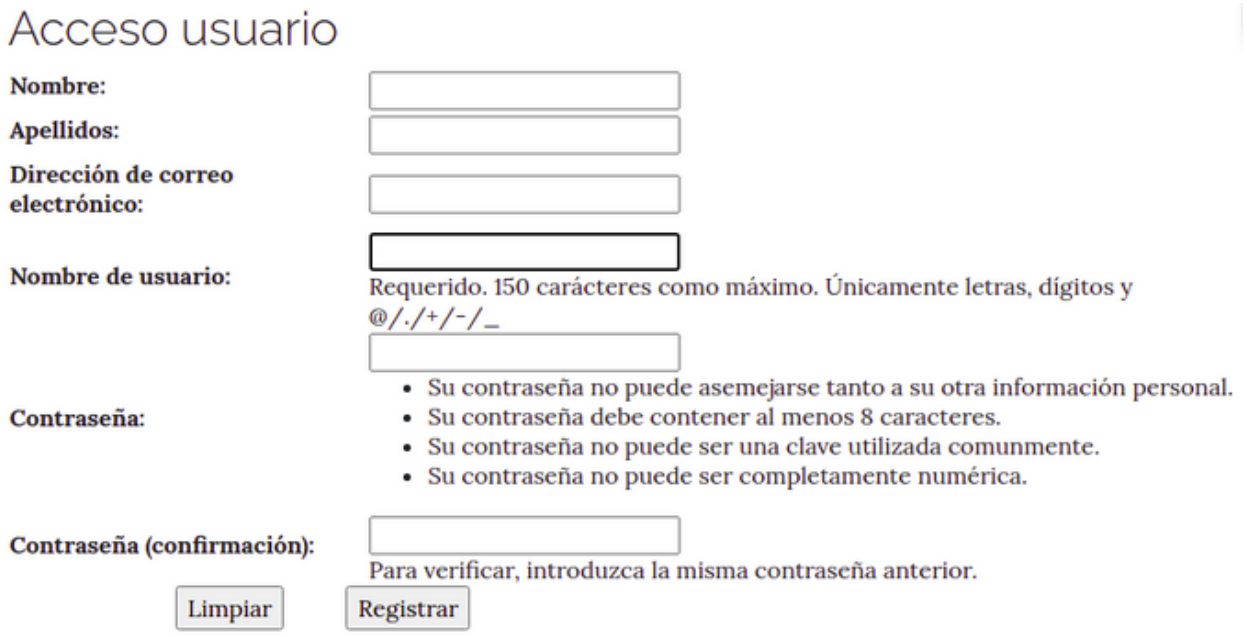
7. Pruebas

Una vez finalizado el desarrollo de la aplicación, se han realizado diversas pruebas para asegurar que tanto el funcionamiento como el manejo de nuestra aplicación web, son los correctos.

Mediante estas pruebas hemos comprobado que la aplicación web cumple con todas las funcionalidades sin errores y con los tiempos de carga apropiados y esperados para la misma. A continuación se explicarán algunas de las pruebas realizadas en la aplicación.

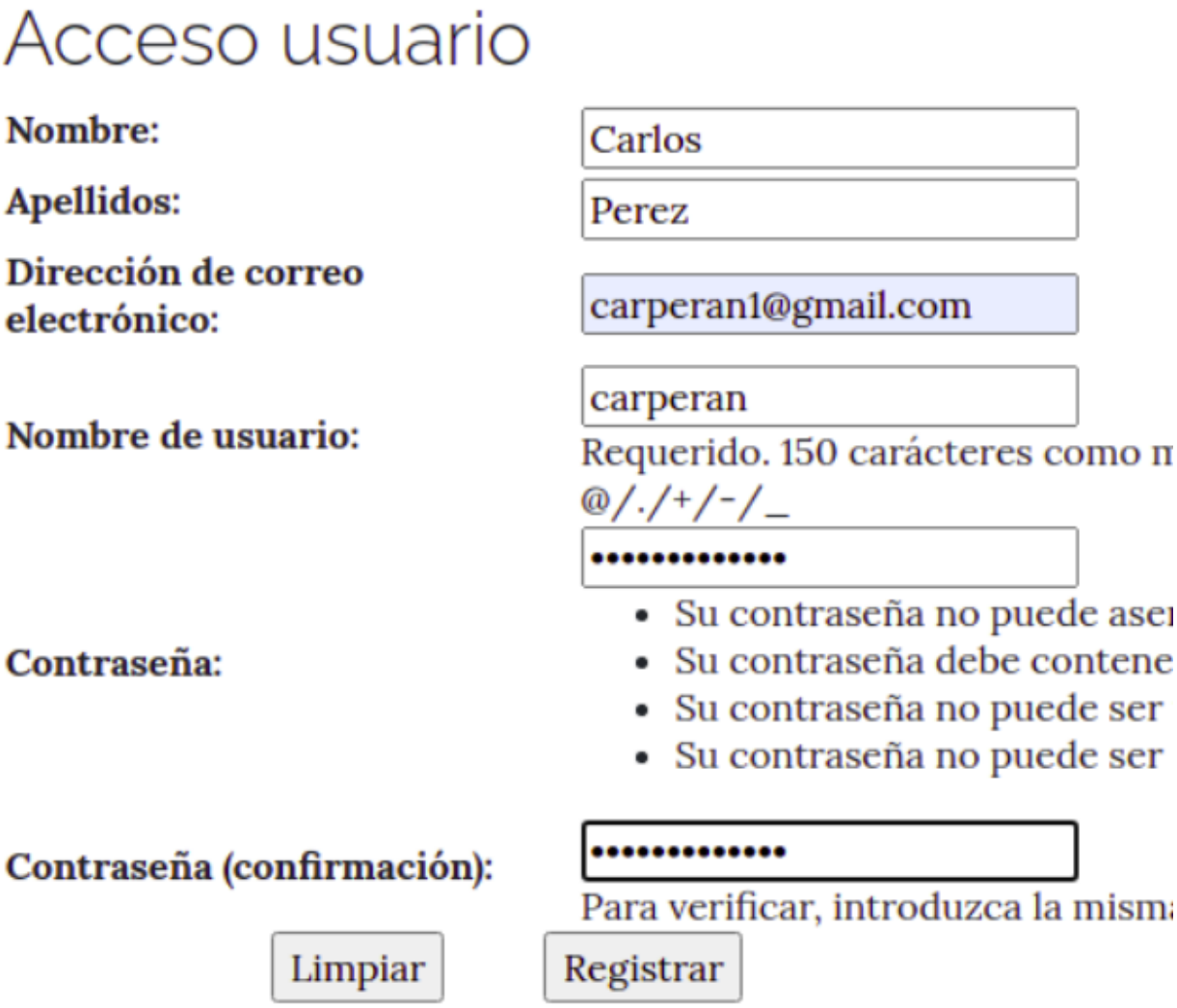
7.1. Pruebas en el registro de la aplicación

Una funcionalidad importante de la aplicación es el registro de los usuarios en la misma, de manera que una vez finalizada la programación de la ventana de registro se realizaron pruebas para probar su funcionamiento.



La ventana que aparece en la imagen anterior es la de registro, una vez rellenada con los datos correctamente se puede proceder por el registro del usuario, una vez que el mismo pulse el botón correspondiente.

También se puede observar que el usuario tiene la opción de resetear todos los campos del formulario de registro, funcionalidad importante en el caso de que algún usuario se equivoque en más de un campo y considere que es preferible borrar todos los campos pulsando un solo botón, a tener que borrar los campos de uno en uno y reescribirlos.

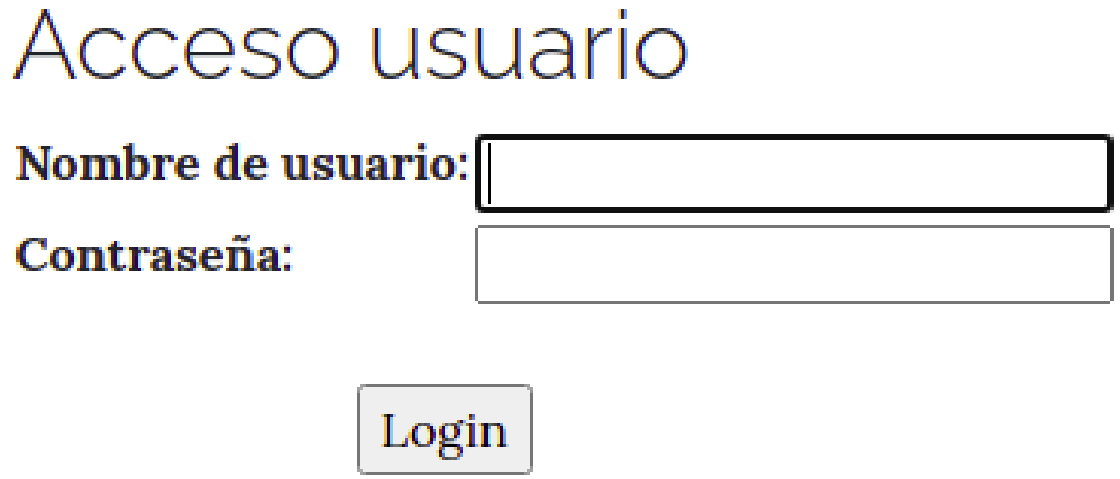


En la imagen anterior podemos observar los campos de la ventana de registro ya rellenados, de manera que se procede a pulsar el botón de registrar usuario para comprobar que la funcionalidad se cumple satisfactoriamente.

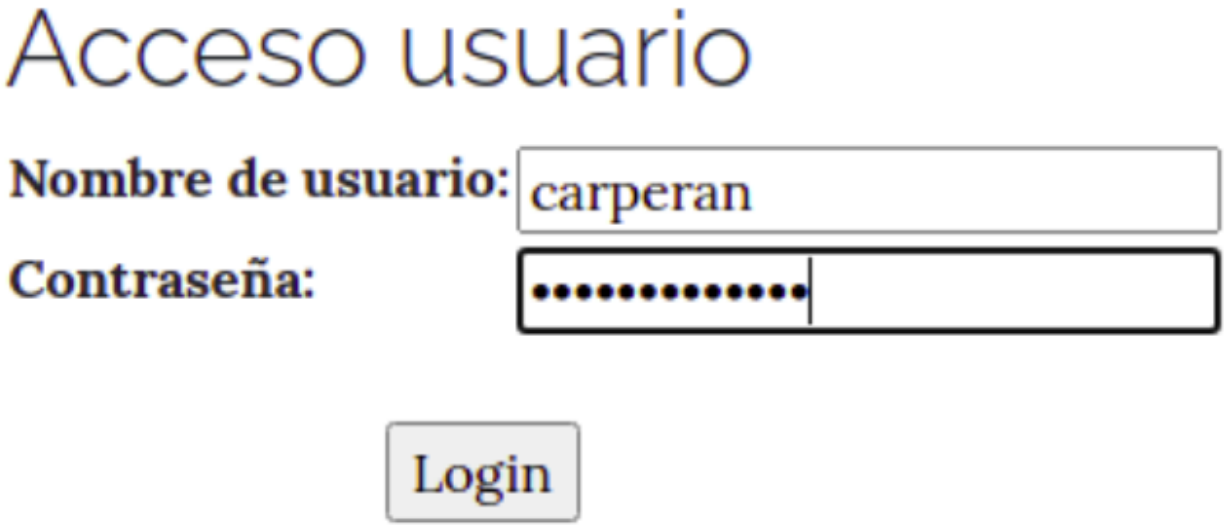
Tal y como se observa en la imagen, desde el panel de administración de Django podemos observar como, a parte del superusuario creado para acceder al mismo, se ha creado el usuario que hemos introducido correctamente, asegurando el correcto funcionamiento de la funcionalidad de registro.

7.2. Pruebas en la autenticación de la aplicación

Otra funcionalidad importante de la aplicación es la autenticación de los usuarios que ya hayan realizado el registro en la aplicación. Una vez la parte de desarrollo de la funcionalidad está terminada, se realizan las pruebas pertinentes para comprobar que funciona de forma correcta.



En la imagen anterior podemos observar la ventana de login de la aplicación. Para comprobar que funciona correctamente se han rellenado los campos con los datos del usuario creado en la prueba de registro mostrada anteriormente.



Como se observa en la imagen anterior, se han rellenado los campos con usuario existente. Para comprobar el funcionamiento de la autenticación solo habrá que pulsar el botón de Login.

En esta imagen se observa como después de pulsar el botón de Login, la aplicación realiza la funcionalidad de autenticación satisfactoriamente, ya que la etiqueta que muestra el nombre de usuario solo se activa cuando hay un usuario autenticado. Esto demuestra que la funcionalidad que se estaba probando funciona correctamente.

7.3. Pruebas en la creación de productos

Una de las funcionalidades más importantes de la aplicación es la de crear actividades y productos ofertados por el centro. En esta prueba vamos a comprobar que una de las dos, en este caso la creación de productos, funciona satisfactoriamente.

Para ello, primero deberemos autenticarnos con un usuario que tenga permisos de administrador, ya que serán los únicos que podrán acceder a estas funcionalidad.

En la imagen anterior observamos la ventana de creación de productos, para comprobar la funcionalidad tendremos que rellenar los campos que aparecen correctamente y pulsar el botón de crear producto. Igual que en la ventana de registrar usuario, tendremos un botón llamado Limpiar, para resetear todos los campos que podemos observar que aparecen en la imagen.

En la imagen anterior observamos los campos de la ventana de crear producto rellenados, solo hay que pulsar el botón correspondiente a crear producto para comprobar que la funcionalidad se ejecuta correctamente.

En la imagen se observa como el producto que hemos introducido anteriormente ha sido creado de manera satisfactoria y con los valores que se han introducido en los campos mostrados en la imagen anterior.

8. Conclusiones

En este apartado me gustaría comentar mis conclusiones con respecto al desarrollo de este proyecto.

Primero, me gustaría mencionar que una gran parte de las horas destinadas al desarrollo de la aplicación han sido dedicadas al aprendizaje de la tecnología empleada en la misma, ya que como se mencionó en el apartado de introducción, es la primera aplicación que he desarrollado utilizando el entorno de desarrollo Django.

Django es un entorno muy completo y potente, si bien al principio puede ser un poco complejo de utilizar, una vez se le dedica tiempo acaba siendo bastante simple, a pesar de no ser la primera vez que utilizo en entorno para desarrollo de aplicaciones web, el funcionamiento de este entorno es bastante diferente al que había utilizado previamente.

Una vez había realizado un proceso de aprendizaje e información de la tecnología a emplear, cuando empezó el desarrollo fue confuso al principio, pero luego acabó siendo bastante intuitivo y para cuando empecé a desarrollar las funcionalidades complejas ya dominaba con cierta soltura el entorno.

A pesar de lo mencionado anteriormente con respecto a la complejidad, estoy bastante satisfecho con el resultado de la aplicación, no porque considere que es de una complejidad desmedida, si no porque creo que pese a ser la primera aplicación que he desarrollado en un nuevo entorno y con lenguajes que no había empleado nunca, el resultado ha sido bastante positivo:

- Las funcionalidades que el cliente ha ido acordando conmigo funcionan correctamente y no tarda excesivo tiempo en ejecutarse.

- Igualmente el cliente ha quedado conforme con el diseño de la aplicación.

- A pesar de los bloqueos a la hora de desarrollar las funcionalidades más complejas de la aplicación, he conseguido que funcionen satisfactoriamente.

Por motivos como los mencionados previamente, estoy contento con el resultado del proyecto. Y personalmente considero que el objetivo personal para desarrollar la aplicación en Django (mencionado en la introducción de la memoria), lo he conseguido; porque he desarrollado una aplicación que ya no solo me parece que tiene un buen diseño y que funciona correctamente, si no que también lo considera el cliente.

Por último, recalcar que a pesar de que en la aplicación desarrollada en este proyecto no se exprime al máximo el potencial de este entorno, mencionar que otras aplicaciones de renombre mundial, como Instagram, hacen uso de este entorno.

9. Bibliografía

[1] Documentación de la página oficial de Django. https://docs.djangoproject.com/en/3.2/.

[2] Documentación de la página oficial de bootstrap. https://getbootstrap.com/.

[3] Nigel George. Mastering django, 2015.

[4] Documentación de la página oficial de scrum. https://www.atlassian.com/es/agile/scrum.

[5] Mark A. Lutz. Learning python: Powerful object-oriented programming, 2013.

[6] William S. Vincent. Django for beginners: Build websites with python and django, 2018.