UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE TECNOLOGÍA

CARRERA DE ELECTRICIDAD INDUSTRIAL

DESARROLLO DEL SISTEMA DE COTIZACIÓN Y ANÁLISIS DE PRECIO UNITARIO PARA INSTALACIONES ELÉCTRICAS

Proyecto de grado Técnico presentado para obtención de Grado de Técnico Superior

POR: CRISTIANO SANTOS DE CARVALHO

TUTOR: M.Sc. JAVIER MARCELO FLORES MONRROY

LA PAZ – BOLIVIA

Junio – 2021

DEDICATORIA

A Dios que me ha fortalecido por todo el camino, a mi familia por su incentivo y comprensión, y a todos los queridos amigos que me apoyaran mucho en esta jornada.

AGRADECIMIENTOS

Mis sinceros agradecimientos a todos los docentes de la carrera de Electricidad Industrial de la Facultad de Tecnología que se empeñaran en entregar su mejor nivel de enseñanza, a mi Tutor, por su suporte incondicional y vasto conocimiento en el tema abordado, a las autoridades académicas por su suporte, comprensión y disponibilidad, a la jefatura de la carrera por su trabajo incansable en dar el mejor apoyo posible a los estudiantes. Finalmente, agradezco a Dios por haber puesto más que docentes y compañeros de carrera, sino amigos que me han enseñado, acogido y recibido como uno de los suyos. Muchas gracias mis hermanos de Bolivia.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

RESUMEN

El presente trabajo propone una solución de integración en el campo de la Electricidad con el desarrollo de un sistema de cotizaciones y análisis de precios unitarios de instalaciones eléctricas. Las tecnologías empleadas para el sistema son desarrolladas para web y comúnmente utilizadas en el mercado de consumo digital para diseño de aplicaciones para la internet.

The present work brings an integrated solution in the Electrical field with the development of a quotation and analysis of per unit price system for electrical installations. The technologies applied to the system are aimed for the web development and broadly used in the digital consume market for designing internet applications.

Capítulo I

Aspectos Generales

1. GENERALIDADES

1.1 INTRODUCCIÓN

Se está viviendo tiempos en que la demanda tecnológica mundial ha convergido de manera vertiginosa, sea por la demanda de automatización de los procesos productivos, por la globalización de la economía mundial o, como en la actualidad, por temas como una pandemia que nos ha lanzado el desafío de reinventarnos para poder seguir con las actividades cotidianas de una forma segura y eficiente.

En este contexto de demanda tecnológica y desafíos modernos, existe una necesidad elemental de que una empresa de servicios eléctricos, este con una buena salud económica para que pueda perdurar en el mercado de consumo. De acuerdo con el informe del sitio web Investopedia (Deane, 2021), el *Bureau of Labour and Statistics* – Departamento del Trabajo y Estadística – en sus datos muestran que aproximadamente un 20% de los nuevos negocios entran en quiebra durante sus primeros dos años de apertura, un 45% durante los primeros cinco años, y que unos 65% durante los primeros 10 años. Apenas 25% de los nuevos negocios llegan a alcanzar los 15 años de vida o más. Tales estadísticas se han mantenido permanentes desde los años 90. Segundo el mismo artículo, el análisis económico está apenas después de la investigación de mercado y contar con un plan de negocios, como razones para las fallas de los emprendimientos.

Podemos indicar que calcular el precio final, poner un valor exacto y bien calculado en un producto o servicio es el punto clave para la sobrevivencia de una empresa a largo plazo. Este concepto no es distinto para el rubro de la electricidad. Por ser un proceso largo de análisis y con muchas variaciones y detalles, la preferencia es saber cuánto está el precio de mercado, sin saber cómo calcular a tales valores.

1.2 ANTECEDENTES

En Bolivia, la empresa ROMSOF ofrece un producto llamado PRESCOM que viene a ser un programa informático *stand alone,* que debe ser instalado en una computadora

mediante la compra de una licencia comercial. Según la descripción del mismo en su sitio web oficial, *http://www.prescom-bo.com/*, indica que está diseñado para la automatización de procesos en el cálculo de precios unitarios, presupuestos, registro de cómputos métricos, cronogramas y seguimiento de obras. Es un software que está en el mercado desde 1988 y está más direccionado al mercado de construcciones civiles.

Con la intención de crear una aplicación que realice los cálculos, de manera específica para el rubro de la Electricidad, encontramos un área inexplorada en el mercado boliviano, dando la posibilidad de encontrar una solución que vaya a suprimir la carencia del mismo.

1.3 PROBLEMA

La falta de un software en el mercado de consumo, que atienda a una demanda en el proceso de presupuestar correctamente, incluyendo los cálculos para hallar el precio unitario con gastos de las especificaciones de la mano de obra, costos administrativos, utilidad deseada y gastos con tasas tributarias como el IVA; además del cálculo específico del rendimiento de la mano de obra en el rubro de la electricidad industrial, es el problema propuesto a ser resuelto en el presente proyecto de grado técnico.

1.4 OBJETIVOS

Los objetivos que definen este proyecto de grado técnico, se dividen en dos categorías, una general donde se enfoca en el producto final a ser diseñado e implementado, y la de objetivos específicos donde se muestra una planificación estratégica para llegar a alcanzar el objetivo general.

A continuación, se describe las dos categorías:

1.4.1 GENERAL

Desarrollar un sistema de cotización y análisis de precio unitario para instalaciones eléctricas a través de una aplicación web, para la planificación financiera de una empresa del rubro eléctrico.

1.4.2 ESPECÍFICOS

- Automatizar el proceso de cálculo del precio unitario con todas sus especificaciones y normas tributarias.
- Implementar herramientas digitales y/o tecnologías de punta que tengan capacidad de expansión, de fácil mantenimiento y estabilidad.
- Utilizar el concepto de *backend* y *frontend* para la implementación del proyecto, donde el primero procesa toda parte lógica, estructural y procedimental, y el segundo, muestra los datos en pantalla;
- Comprobar las funcionalidades programadas, utilizando como base los conceptos de cálculo de precios unitarios.

1.5 JUSTIFICACIÓN

En términos de viabilidad del proyecto, se presentan las siguientes justificaciones: económica, social, tecnológica y académica.

1.5.1 ECONÓMICA

Una de las principales causas de quiebra de un negocio o empresa, es por causas económicas. Ya sea la inyección de capital, una mala administración de la misma, pérdidas económicas, cambio de mercados y crisis. Todos estos factores afectan directamente la estabilidad económica de un emprendimiento.

Saber exactamente cuánto cuesta a una empresa prestar un servicio o proveer al mercado un producto, asegura la competitividad y supervivencia de la misma a lo largo

de los años. Un negocio que no sabe calcular sus precios debidamente puede caer en el error de sufrir pérdidas o decremento de capital, por pagar impuestos o costos que no se han incorporado como sus costos de producción, sean fijos o variables. Además, se puede incurrir en el error de sobrevalorar los precios y tener un impacto fatal al no contar con precios competitivos en el mercado.

Por lo tanto, este proyecto pretende asegurar la posibilidad de competitividad, rentabilidad y estabilidad financiera a través del cálculo de precio unitario.

1.5.2 SOCIAL

La importancia de un buen análisis de cálculo se extiende al ámbito social. Como la estabilidad económica de un negocio está directamente unida a la venta de productos y/o servicios, los daños causados para una empresa que no logra saber cuánto son sus gastos de producción o prestación de servicios pueden llevarla a su quiebra.

La bancarrota de un negocio además de representar una menor tasa de empleos y menor poder de compra, también representa una menor oferta de productos y servicios a la sociedad, pudiendo impactar directamente en su calidad de vida.

Cuanto más saludable y enérgica se encuentre una empresa, se encuentran mayores oportunidades de trabajo, crecimiento económico, recaudación de impuestos, beneficios sociales y la posibilidad de un crecimiento en la calidad de vida de su población.

Por lo tanto, es esencial para un negocio, especialmente para el rubro de servicios en electricidad, tener la capacidad de calcular de manera apropiada sus precios de productos y servicios.

1.5.3 TECNOLÓGICA

Con la creciente demanda de las actividades digitales, el área académica está cada día más desafiado a presentar una solución de integración entre lo académico y sus beneficios e impactos que pueden traer sobre la sociedad y mercado de consumo.

Como una forma de expresar esta integración, este proyecto propone desarrollar para los profesionales eléctricos, la aplicación de software práctico para el cálculo de precios unitarios en un forma sencilla, actual y basada en tecnologías modernas que están en la vanguarda del desarrollo web. El proyecto es una integración moderna de los conceptos académicos con lo mejor de la tecnología moderna para *web framework.*

1.5.4 ACADÉMICA

El presente proyecto de grado técnico hace una combinación práctica entre temas teóricos como Costos y Presupuestos, Emprendimientos Empresariales, Instalaciones Eléctricas, entre otros, y de igual forma, una conexión entre las más recientes tecnologías de punta respecto a aplicaciones web, posibilitando un proyecto de relación interdisciplinar que pueda proporcionar una expansión en la gama del campo de investigación dentro de la carrera de Electricidad Industrial. Además, se proporciona información que brinda la posibilidad de desarrollar software para migrar a un ambiente digital en el área de electricidad industrial.

1.6 ALCANCES Y LÍMITES

Para tener una comprensión más precisa de lo que abarca el proyecto, vamos a definir el alcance del mismo, la lógica que está incluida en sus funcionalidades se limita a la aplicabilidad del fundamento teórico del costo unitario en el rubro eléctrico.

1.6.1 ALCANCES

Los alcances del presente proyecto se enmarcan en la elaboración de un documento técnico e informativo sobre la implementación de una aplicación web en la cuál en su

punto principal se hará un análisis de precios unitarios que posibilite determinar el costo por unidad de servicios en el área de electricidad de forma automatizada. En su contenido podemos citar:

- Ingreso de Clientes;
- Registro de materiales;
- Registro de Mano de Obra;
- Registro de Rendimientos;
- Registro y cálculo de Precios Unitarios;
- Registro de Cotizaciones;

1.6.2 LÍMITES

El proyecto se limita a implementar una versión beta del Sistema de Cotización y Análisis de Precio Unitario. El concepto de una versión beta es que el sistema no está en producción, por lo tanto, tiene áreas que necesitan mayor desarrollo e implementación. El alcance del proyecto no incluye todas las funcionalidades de una aplicación en producción y servicio comercial que cuenta con un equipo informático de tiempo integral. Sus limitaciones se listan como:

- No se ha creado vistas para borrar o actualizar los registros, aunque dichas funcionalidades están implementadas en el *backend* de la aplicación;
- Como el enfoque del proyecto es el análisis de costos para servicios eléctricos, el registro de clientes se ha limitado a un campo de datos: nombre;
- El ingreso de datos para la Mano de Obra igualmente se limita a campos como Cargo y Salario, por no necesitar mayores informaciones respecto a mano de obra para los cálculos de servicios de instalación eléctrica domiciliaria;
- Todas las vistas están diseñadas utilizando el formato CSS (*cascade style sheet*) que es utilizado por medio del Bootstrap de forma minimalista en su apariencia,

sabiendo que el enfoque del proyecto es la funcionalidad y demonstración de los conceptos teóricos aplicados;

- Para el registro de los Rendimientos utilizados en el análisis de Precios Unitarios, se presupone cálculos previos del mismo por no estar incluido en el objetivo de la aplicación, un análisis de tiempos y cálculos de rendimientos;
- La capacidad de ingreso de materiales para el cálculo de precios unitarios se ha limitado en diez ítems para efecto demostrativo de los conceptos;
- De la misma manera, la capacidad de ingreso de materiales por cotización se a limitado a diez ítems;
- Aunque para acceder al sistema se hace necesario la utilización de un usuario y contraseña, implementado en el *backend*, no se ha realizado de la misma manera en el *frontend*, el cual cuenta con una vista específica para la administración de registro de usuarios y contraseñas;
- Para el ingreso de datos al sistema, se asume que el operador del sistema tiene una lista de rendimientos para actividades o servicios previamente calculados, como se poder ver en el anexo. Por ejemplo, como una lista de materiales que necesitan ser implementados en un proyecto de una vivienda, departamentos, y otros.

1.7 METODOLOGÍA

La metodología empleada para la solución del problema propuesto, es por medio de la investigación de tecnologías de punta que están siendo utilizadas actualmente y que tienen una capacidad de planificación por demanda, vasta utilización, facilidad de mantenimiento y mano de obra disponible. Su aplicabilidad utilizará el método experimental de desarrollo de software, mediante análisis de datos, normalización de tablas y base de datos, implementación de rutinas, procedimientos, código de programación, diseño de interface gráfica, con pruebas de funcionamiento y cumplimento de los objetivos proyectados.

Capítulo II

Marco Teórico

2. MARCO TEÓRICO

2.1 INTRODUCCIÓN

La ciencia de la administración, ha evolucionado al largo de los años, brindando vasta cantidad de material sobre gestión empresarial, y nociones académicas para que la tarea de administrar un negocio sea más entendible y con mayores oportunidades de éxito.

Toto emprendimiento, en cualquiera rubro aplica esta ciencia en alguna medida, aún que no de manera directa y que muchos emprendedores no sepan de conceptos teóricos. Cuando tratamos el tema de Costos y Presupuestos, de igual manera estamos tratando de temas que hasta cierta forma son empíricos a las personas que ofrecen sus productos y servicios al mercado. Nuestro objetivo es lanzar bases fundamentales en términos teóricos para dar apoyo al presente proyecto.

Primeramente, se describirá algunos conceptos importantes respecto al proceso productivo de bienes y servicios, detallando conceptos que se va a aplicar en todo el proceso, hasta que se pueda hallar el costo final de los mismos.

2.2 EFICIENCIA Y EFICACIA

Segundo Óscar Heredia Vargas (Heredia 2007), la eficacia está relacionado a cumplir una tarea, logar concluir una tarea deseada, finalizarla. Por ejemplo, la eficacia seria producir una cantidad de productos determinada.

La eficiencia es una relación entre recursos y productos, donde si tiene los recursos necesarios y a través de su manipulación se obtienen los resultados. En suma, sería aplicar una metodología o secuencias de procesos para llegar a un determinado fin. Cumplir un objetivo, para lograr un producto final. Por lo tanto, se debe utilizar de la forma más optima posible los recursos disponibles, producir lo máximo posible, utilizando el mínimo número de recursos.

Consiguientemente, una empresa es eficaz cuando realiza y termina una tarea produciendo un resultado a partir de los recursos que posee, mientras que es eficiente cuanto utiliza óptimamente los recursos con el mayor provecho posible de ellos.

2.3 COSTO VARIABLE

Para el autor del libro Administración de Operaciones, (Krajewski, 2008), el costo variable es la parte del costo total que varía directamente con el volumen de producción. Este costo está relacionado a la cantidad de material necesaria, mano de obra, y todos los gastos asociados con el proceso productivo. Por lo tanto, cuanto mayor el volumen de producción, dependiendo de los gastos son derivativos del proceso productivo, mayor su variación.

2.4 COSTO FIJO

Los costos fijos son los costos que no cambian con el proceso productivo, o sea, no están directamente relacionado a la producción. Si el volumen de producción aumenta o disminuye, los costos fijos, como el propio nombre mismo lo sugiere, permanece igual. Su impacto en el precio final es que, al aumentar el volumen de producción, como tales costos se mantienen constantes, es menor a mayor volumen productivo.

2.5 PUNTO DE EQUILIBRIO

El punto de equilibrio es la cantidad necesaria en términos de volumen de producción para que se igualen los valores de ingreso a los valore totales de la producción. Toda cantidad producida por debajo de este punto se refleja en pérdidas, mientras que, los valores por encima indican ganancia sobre los ingresos.

Es decir que el análisis del punto de equilibrio indica cuál es la cantidad mínima que se debe producir para que pueda tener ganancia o pérdida en los procesos productivos o en la prestación de servicios.

Para determinar el punto de equilibrio, es necesario saber en primero lugar cual es el valor total de un producto producido y su fórmula está dada en función del precio fijo (P_F), del precio variable (P_V) y la cantidad del resultado del proceso productivo (Q), observamos por la ecuación que el costo total es una ecuación lineal. Tenemos que:

$$Costo\ total\ = P_F + P_V * Q \cdots (1)$$

Además, se necesita saber el ingreso total de dicha producción, asumiendo que sea completamente vendida, para que se pueda analizar cuando los ingresos se igualan al costo total de producción. Por lo tanto, se tiene que tomar en base al precio unitario de venta (P_U) de la cantidad total producida. La fórmula de cálculo de los ingresos es:

$$Ingreso\ total = P_U * Q \cdots (2)$$

Considerando que conceptualmente el punto de equilibrio es la igualdad entre ingresos y costo total de producción, tenemos:

$$Igualando\ (1)\ y\ (2):$$

$$P_U * Q = P_F + P_V * Q\ \therefore\ Q = \frac{P_F}{P_U - P_V} \cdots (3)$$

En el caso de la ecuación (3), Q representa el punto de equilibrio en lo cual no hay ni pérdidas, ni ganancias.

2.6 ¿POR QUÉ SE DEBE SABER CUÁNTO CUESTA UN PRODUCTO O SERVICIO?

La importancia de saber cuánto cuesta un producto o servicio es poder determinar algunos aspectos sobre el mismo, como por ejemplo si el producto es competitivo en el mercado de consumo, si la cantidad producida está de acuerdo a las expectativas de ganancia. Además de todo lo mencionado, lo más importante es saber cuándo se está teniendo pérdida y como poder tomar decisiones para incrementar la rentabilidad del producto en el mercado sin afectar la competitividad del mismo.

Como se ha visto, una de las razones para el fracaso de un negocio es la parte económica. No saber cuánto cuesta producir o prestar un servicio, es no saber si hay o no perdidas; que potencialmente puede llevar el emprendimiento a su fin. Trasfiriendo este conocimiento al rubro de la electricidad, está claro que la actividad final no es la venta de producto, sino la prestación de un servicio que, muchas veces, puede incluir o estar asociado con el suministro del mismo, como parte del servicio.

2.7 ¿CÓMO SE PUEDE CALCULAR EL PRECIO UNITARIO DE UN SERVICIO DEL RUBRO ELÉCTRICO?

Similarmente que vender un producto, para la prestación de un servicio se necesita saber exactamente cuánto cuesta realizar determinada tarea, solamente con el conocimiento de este valor es que se puede garantizar que el negocio sea rentable y estable económicamente. Para el cálculo del precio unitario de un servicio que incluye material, se considerará los siguientes factores:

- Precio del material empleado en el servicio;
- Valores de la mano de obra para la elaboración del dicho servicio;
- Gastos administrativos generados por la prestación del mismo;
- Desgaste del equipamiento para su ejecución;
- Equipos Personales de seguridad;

- Utilidad deseada;
- El Impuesto sobre Valor Agregado estipulado por ley sobre servicios y productos;

Se procederá de la siguiente forma. Primeramente, se realizará una lista de los materiales necesarios para prestación del servicio con sus valores unitarios y respectivas cantidades. Luego, listamos cual será la mano de obra necesaria para la actividad descrita. Acá hay que considerar que se incluya un jefe de cuadrilla que va a supervisar y garantizar la calidad del servicio.

Para los valores de herramientas y equipos de seguridad se considerará una adición de 2% y 4% respectivamente incidiendo sobre el valor total de la suma de la mano de obra. A es valor se suman los valores del material y de mano de obra para luego añadir un porcentaje de gastos administrativos que consideraremos como un 10%. La fórmula para herramientas y equipos de seguridad será:

$$total_{herramientas} = \frac{total_{mo}}{(1 - \%_{herramientas})} \cdots (4);$$

$$total_{seguridad} = \frac{total_{mo}}{\left(1 - \%_{seguridad}\right)} \cdots (5);$$

La fórmula utilizada para agregar los gastos administrativos es:

$$total_{gasyos} = \frac{total_{m.o.} + total_{material}}{(1 - gastos)} \cdots (6)$$

Luego, se calcula sobre el precio el porcentaje de la ganancia deseada, para finalmente calcular el valor tributario del IVA con un porcentaje del 13% (tasa nominal), llegando al precio final.

Es importante resaltar que, para agregar el valor del IVA, utilizaremos la tasa efectiva (T_E) para agregar de manera correcta el gravamen tributario al precio final unitario deseado. Para hallar la tasa efectiva a partir de la tasa nominal (T_N) utilizaremos la siguiente fórmula:

$$T_E = \frac{T_N}{1 - T_N} * 100 \cdots (7) \therefore T_E = \frac{0{,}13}{1 - 0{,}13} * 100 = T_E = 14{,}94\%$$

Para hallar el valor bruto final utilizaremos:

$$Precio_{Neto} = total_{material} + total_{mo} + total_{herramientas} + total_{seguridad} + total_{encargos}$$

$$Valor_{bruto} = Precio_{Neto} * T_{N(IVA)} \cdots (8)$$

Teniendo los precios unitarios ya calculados, se puede hacer con seguridad una cotización que refleje la realidad de todos los gastos incluidos en el precio final. Cuando al fin del año fiscal el emprendedor necesite pagar sus obligaciones tributarias con el gobierno, no deberá descontar nada de sus utilidades.

Capítulo III

Aspectos del Proyecto

3. DESARROLLO DEL PROYECTO

3.1 DESCRIPCIÓN GENERAL DEL SISTEMA

El Sistema de Cotización y Análisis de Precio Unitario, está implementado con una unión de tecnologías que harán posible su concepción y desarrollo. Se realizará la descripción de tecnologías como Django 3.2.4, PostgreSQL 13, Python 3.9.5 y VSCode.

Para su base de datos se debe:

- Proyectar una estructura de datos que responda a las siguientes entradas:
  - Clientes – solamente un campo de datos [nombre];
  - Materiales – con los campos de datos [nombre, marca, descripción, precio, unidad];
  - Mano de Obra – con campos para almacenar dados de [Cargo, Salario];
  - Rendimientos – sus campos de datos son [nombre, descripción, valor de unidades por jornal];
  - Precios Unitarios – tabla de datos que se relaciona con Materiales, Mano de Obra y Rendimientos con los campos calculados [valor de materiales, valor de mano de obra], el relacionamiento con las demás estructuras de datos se da a través de llaves extrajeras *foreign keys*;
  - Cotizaciones – que se relaciona con la tabla de Precios Unitarios, y tiene como campos [Cantidad, valor de materiales, valor de mano de obra, fecha en que fue creada];

3.2 TECNOLOGÍAS EMPLEADAS

3.2.1 DJANGO WEB FRAMEWORK

Django es una estructura de trabajo para desarrollo de aplicaciones web gratuita de código abierto, en inglés conocido como *"open source web framework",* que está basada

en el lenguaje de alto nivel conocida como Python. Es un conjunto de paquetes de clases con las más variadas funcionalidades escritas en Python que fueran desarrollados al largo de casi dos décadas de existencia.

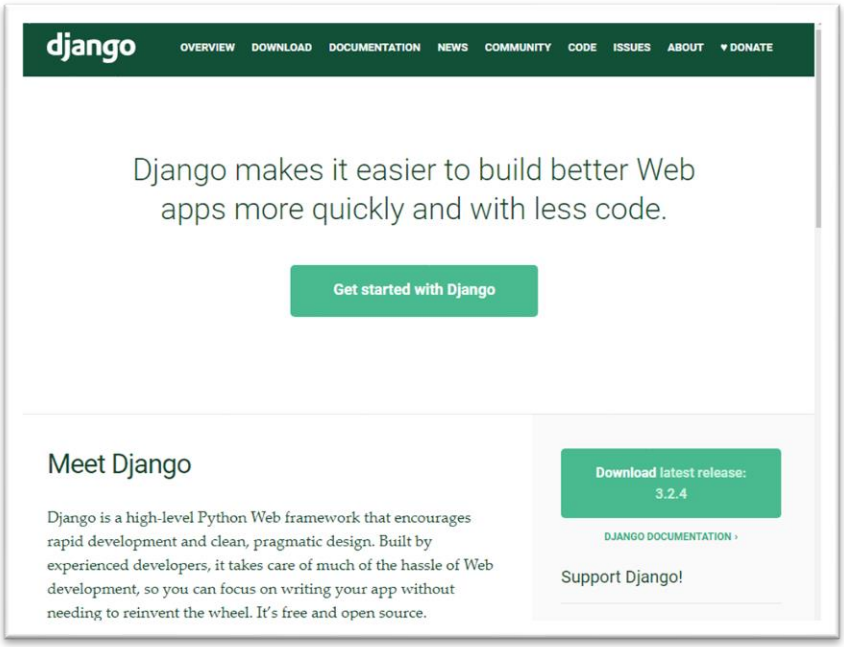


Figura 1 - Sitio oficial del Django.
Traducción propia: Django construye de manera más fácil y rápida mejores aplicaciones para web y con menos código.
Fuente: www.djangoproject.com

Este *framework* que está actualmente en su versión 3.2, es ampliamente utilizado en estos días por reducir la cantidad de procedimientos y/o tareas que un desarrollador web normalmente tiene que seguir. Por esta razón, sitios como Pinterest, PBS, Instagram, Washington Times, Mozilla, y muchos otros más han escogido Django como su plataforma de desarrollo.

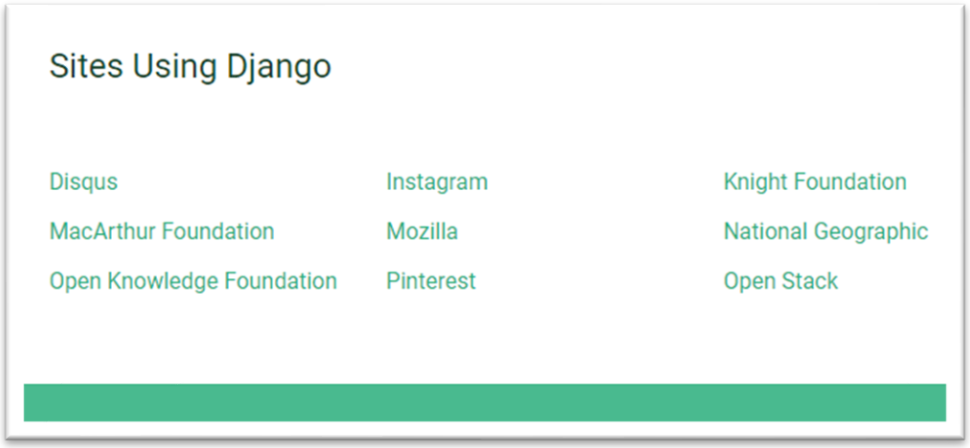


Figura 2 - Sitos web que utilizan el framework Django.
Fuente: https://www.djangoproject.com/start/overview/

Uno de los beneficios más significativos es no tener un gran conocimiento sobre lenguaje de programación para comunicarse con las bases de datos SQL (*Structured Query Language* – en castellano: Lenguaje de Consulta Estructurada). En el caso del Django, la capa de la base de datos está encapsulada y el programador solamente tiene que escribir una clase que representa la estructura de datos deseada, además Django se encarga de comunicarse con la base de datos y crear toda la estructura de datos según las especificaciones de la clase creada sin que el programador tenga que acceder directamente al banco de datos.

Otra característica importante de este *framework*, es su documentación de extrema calidad, extensamente detallada y con una larga gama de ejemplos de cómo se implementan cada ítem descrito. Esto es debido al hecho de que surgió en una redacción de un periódico en los Estados Unidos y los escritores hicieron un gran trabajo documentando Django, sus componentes, funcionalidades y sintaxis.

3.2.2 POSTGRESQL BASE DE DATOS

El PostgreSQL es un base de datos gratuita de código abierto que tiene como características, transacciones confiables y seguras, actualizaciones automáticas de vistas, *triggers*, llaves extranjeras y funciones almacenadas. Posee la capacidad tanto para componer aplicaciones *stand alone* (para la PC), como para *data warehouses* o *web services* con un gran número de usuarios. La versión más actual estable es el PostgreSQL 13.

Figura 3 - PostgreSQL sitio oficial.
Traducción propia: "PostgreSQL: la base de datos relacional más avanzado del mundo".
Fuente: www.postgresql.org

Está disponible para plataformas de sistemas operativos macOS, Windows, Linux, FreeBSD y OpenBSD. El PostgreSQL posee integración con el *web framework* Django que puede conectarse al mismo para crear, actualizar, leer y borrar datos desde las bases creadas.

Otra gran ventaja de utilizar el PostgreSQL es su sistema de administración de la base de datos, una aplicación que se puede utilizar para actualizar campos, hacer pruebas y consultas. Durante el proceso de desarrollo del proyecto fue una herramienta fundamental para garantizar la integridad de los datos y verificar compatibilidad de tipos entre otros.

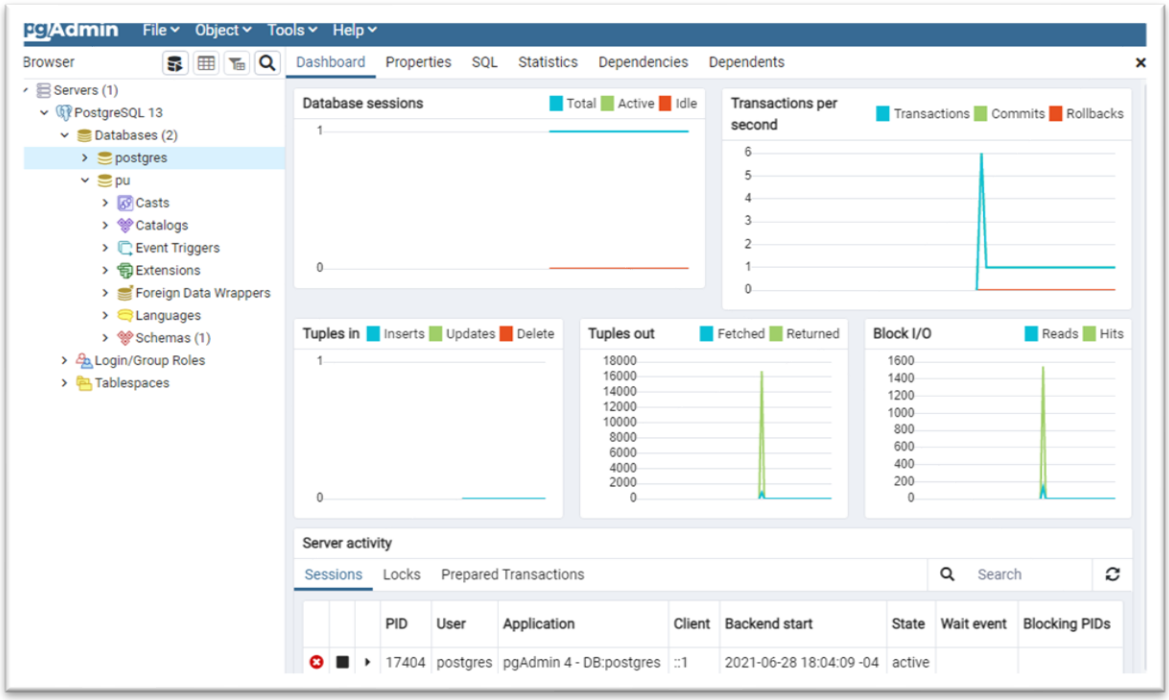


Figura 4 - Sistema de administración - PostgreSQL.

En la lista de base de datos, "pu" es la base creada para soportarla estructura de datos del presente proyecto.

Fuente: propia.

La ventaja de este banco de datos está dada por la posibilidad que tiene de adaptación a los cambios y escalonamiento de la demanda de procesamiento de datos y posibilidad de futura expansión. Desarrollo de funcionalidades y alcance de distintos tares en el rubro eléctrico, que no sería posible con la base de datos estándar utilizada por Django, el sqlite.

3.2.3 PYTHON

Python es un lenguaje de programación de alto nivel, entre las más utilizadas en el planeta. Ha experimentado un creciente número de usuarios cada año y, aunque cuente con tres décadas de existencia, sigue ganando cada vez más espacio en todas las áreas de aplicaciones.

Figura 5 - Sitio oficial del Python.

Traducción: Python es un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional

Fuente: www.python.org.

El enfoque de este lenguaje es la presentación y facilidad de lectura de código. No utiliza, como muchos otros lenguajes, el cierre de línea. El interpretador Python reconoce la tabulación o espacios blancos siendo el alcance de los bloques de códigos contenidos en sus respectivas clases.

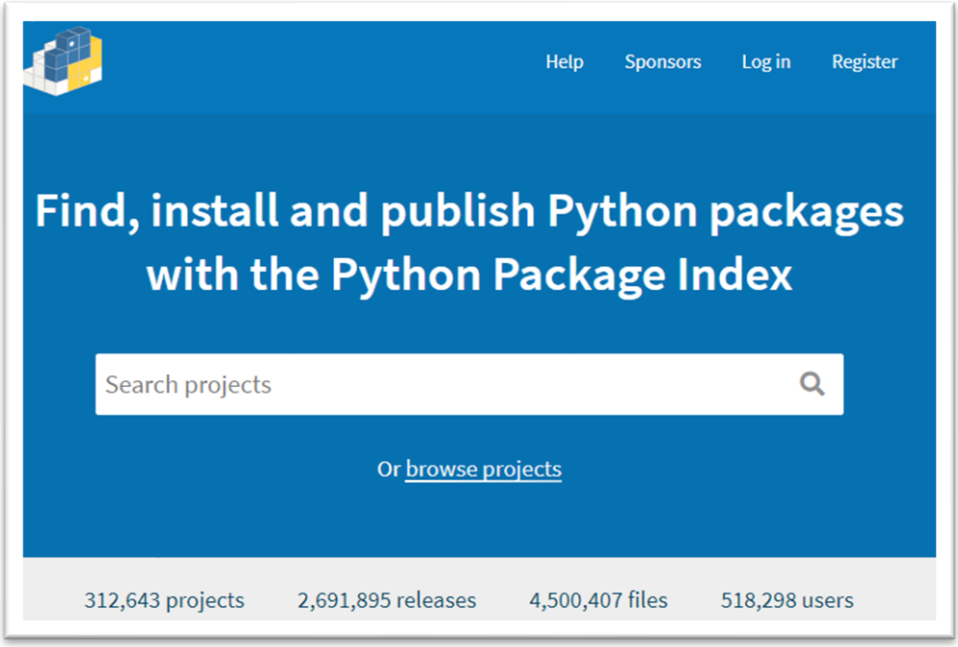


Figura 6 - Busca de paquetes hechos y publicados en Python por los usuarios.
Fuente: https://pypi.org/

La sintaxis de Python es relativamente simple y puede ser utilizada para programación estructurada u orientada a objeto. Un área que este lenguaje ha ganado bastante campo en los recientes años, es el procesamiento de gran volumen de datos o el apoyo a los cálculos científicos.

3.2.4 EDITOR DE CÓDIGO VISUAL STUDIO CODE (VSCODE)

Para el desarrollo de la aplicación se necesita un software en el cual se pueda escribir el código de programación, el llamado editor de códigos. El editor de códigos posee algunas características especiales que facilitan la tarea del programador como:

- Esquemas de colores para las vistas;
- Esquema de colores para código con resaltador de sintaxis;
- Iconos distintos para los diferentes tipos de archivos que se puede confeccionar en el mismo;

- Explorador de ficheros y archivos que facilita la navegación por los componentes del proyecto;
- Capacidad de búsqueda;
- Depuración de código;
- Administración de extensiones entre otras;

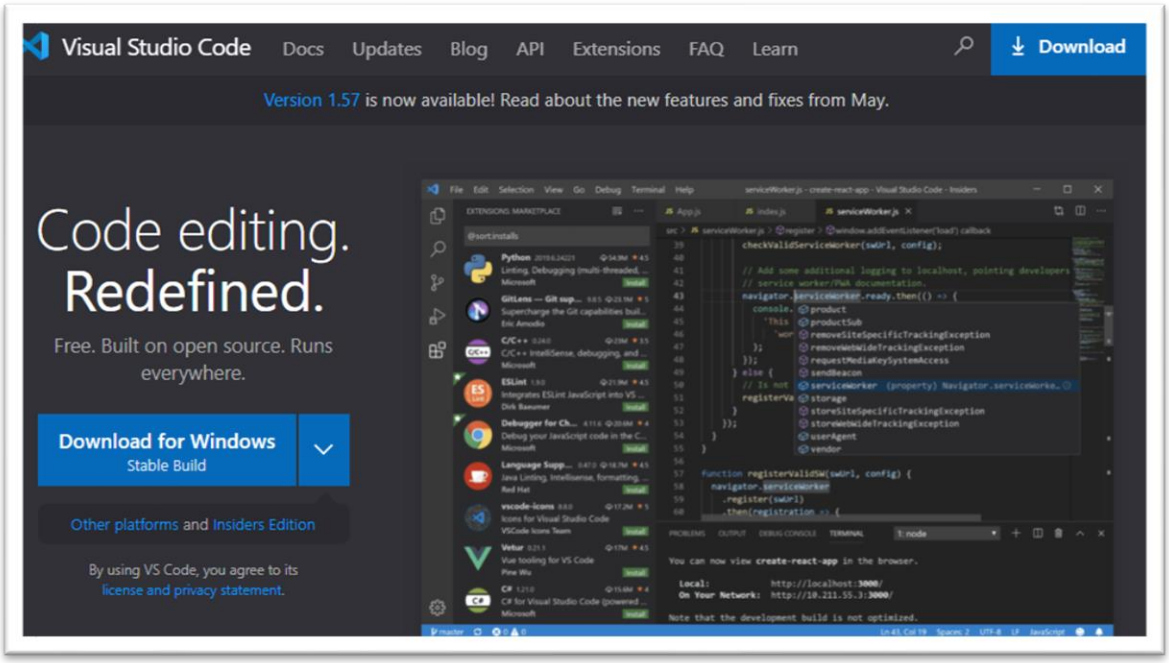


Figura 7 - VSCode, editor de códigos de la Microsoft.
Fuente: https://code.visualstudio.com/

3.3 IMPLEMENTACIÓN

La implementación de este proyecto de grado técnico, sigue los prerrequisitos de las tecnologías seleccionadas, por lo tanto, es necesaria una breve aclaración de como las partes individuales interactúan entre sí para formar una aplicación más robusta y compleja.

3.3.1 PYTHON

El centro de todo el proyecto gira alrededor de un lenguaje principal en que el *web framework* fue escrito, en este caso es el Python. Todas las declaraciones de clases, métodos, procedimiento y sintaxis, siguen el patrón estipulado por Python. Como se puede observar en la figura de abajo, se hace uso de la tabulación para determinar los bloques de código y a que línea pertenece. Por lo tanto, todas las líneas que tienen sangría hacia adentro después de la declaración de una clase, pertenecen al bloque de dicha clase.

```
class MaterialesForm(forms.ModelForm):
    class Meta():
        model = Materiales
        fields = '__all__'
        labels = {
            'material_nombre': '',
            'marca': '',
            'descripcion': '',
            'precio': '',
            'unidad': '',
        }
        widgets = {
            'material_nombre': forms.TextInput(attrs={'class': 'form-control',
                                                      'placeholder': 'Material - Nome'}),
            'marca': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Ingrese la Marca'}),
            'descripcion': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Descripción'}),
            'precio': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Precio del Material'}),
            'unidad': forms.Select(attrs={'class': 'form-control'}),
        }
```

Figura 8 - Código del lenguaje Python en el editor VSCode.
Fuente: Elaboración propia.

Python tiene una sintaxis considerablemente sencilla, a diferencia de otros lenguajes de programación web. Es considerado uno de los lenguajes más utilizados en el aprendizaje de lógica de programación. En ambientes académicos, la gran mayoría de las Universidades, incluso la UMSA, la utilizan por su sencillez y aspecto visual para aprender a programar. Como ya se mencionó anteriormente Django fue enteramente escrito en Python.

3.3.2 DJANGO

Django es el alma de desarrollo de este proyecto de grado técnico. Se puede entender que es un conjunto de paquetes que facilitan las acciones necesarias para implementar un sitio web, un servicio web, entre otros. De ahí viene el significado de un *web framework*, que pode ser traducido literalmente como un marco, una estructura de trabajo para desarrollo web.

En su librería de paquetes, Django incluye una gran variedad de recursos, listos para ser utilizados sin que se tenga que codificar líneas y líneas de código. Como, por ejemplo, Django incluye la funcionalidad *out of the box* (recién sacado de la caja):

- Autenticación de usuarios;
- Pruebas;
- Modelos de base de datos, formularios, URLs y plantillas todos listos para utilizarlos;
- Interface para administración (*backend*);
- Actualizaciones de seguridad y performance;
- Suporte para una gama variada de base de datos;

La forma en que fue diseñado, es para trabajar con la concepción de MTV – *Models, Templates y Views* (Modelos, Plantillas y Vistas) donde el modelo es la base de datos o un mapa de los relacionamientos de los datos, las plantillas son los formas que se utilizarán para que puede imprimir por pantalla la lógica de negocio determinada por las vistas. Podemos ver en la figura 9 como se relacionan las partes conceptuales del Django y cuáles son las informaciones que cada una de ellas administra o proporciona al esquema como un todo.

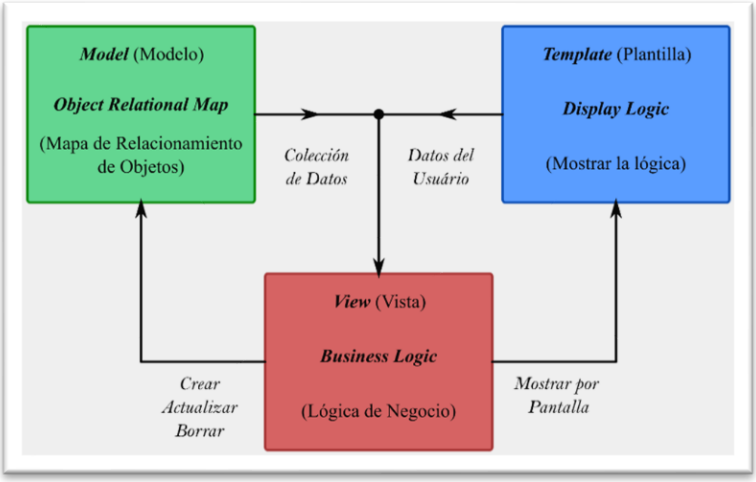


Figura 9 - Estructura conceptual de Funcionamiento del Django.
Fuente: propia.

Como podemos observar en el diagrama de la figura 9, el modelo brinda a la vista con un conjunto de datos al cual se aplicará una serie planificada de pasos, una lógica de negocios, que puede ser almacenada, actualizada o borrada de la base de datos. De igual manera podemos observar que la plantilla es utilizada para recolectar datos ingresados por el usuario que de igual manera van ser procesado para que sean presentados en pantalla. Se mostrará este comportamiento de forma práctica en el funcionamiento del sistema.

Estos tres elementos forman, como se dijo anteriormente, el alma del sistema. En esto sentido, los modelos acoplan el Django con el PostgreSQL, las vistas del centro de procesamiento de datos que es la integración con Python, y finalmente las plantillas conectan el *backend* a la capa de visualización, el mencionado *frontend*.

La estructura de archivos de una aplicación web implementada con estas tecnologías, es dada por un fichero principal que es considerado el paquete principal del sistema, una *app* que, a pesar de su nombre, se comporta más como un componente de funcionalidad que una aplicación en separado, esta *app* tiene un fichero propio, y adentro de la carpeta de la *app* se localizan los archivos de Python, la carpeta de las plantillas, y otros.

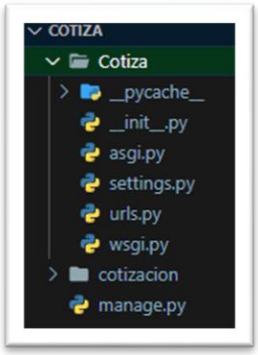


Figura 10 - Estructura del Proyecto.

Fuente: Propia.

En la figura 10 podemos observar la estructura Django que se crea cuando se coloca el comando en el terminal `django-admin startproject <nombre del proyecto>`. El próximo paso sería crear la *app*, con el comando `djangopython manage.py startapp <nombre de la app>`, obteniendo la estructura de datos observada en la figura 11.

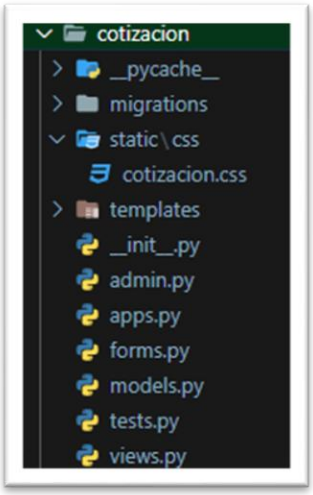


Figura 11 - Estructura de la app cotización.

Fuente: Propia.

Adentro de la carpeta *templates* se encuentran los archivos en extensión *html*, que son las plantillas para mostrar la pantalla de datos.

3.3.3 ANÁLISIS DE PRECIO UNITARIO

Para el Análisis de Precio Unitario en el contexto de del desarrollo del sistema, se entenderá como el proceso en que se puede agregar una cantidad de material o materiales a la base de datos, valores de la mano de obra, el rendimiento de las actividades correspondientes a instalaciones eléctricas y las tasas para calcular los costos adicionales del desgaste de herramientas, equipos personales de protección, los gastos administrativos, la utilidad deseada y finalmente, el impuesto sobre el valor agregado (IVA).

El modelo de la tabla de precios unitarios es declarado en el archivo *models.py*, que como dicho anteriormente es una abstracción de la base de datos. Los atributos del objeto que pertenece a esta clase son el rendimiento, una llave extranjera que hace la conexión entre los precios unitarios y rendimientos, material_id, la conexión entre los materiales y precios unitarios, mano_obra otra conexión entre modelos, valor_mo y valor_material que son campos calculados cuando se graba un registro en la base de datos.

```
class PreciosUnitarios(models.Model):
    rendimiento = models.ForeignKey(
        Rendimientos, on_delete=models.CASCADE, related_name='rendimiento')
    mano_obra = models.ManyToManyField(
        ManoObra, through='PuManoObra', related_name='mano_obra')
    material_id = models.ManyToManyField(
        Materiales, through='PuMaterial', related_name='material')
    valor_mo = models.FloatField(
        blank=True, null=True)
    valor_material = models.FloatField(
        blank=True, null=True)
```

Figura 12 - clase "PreciosUnitarios".

Fuente: Propia.

```python
def save(self, *args, **kwargs):
    if self.id:
        pk = self.id
        v = Variables.objects.get(pk=2)
        rm = (self.rendimiento.unidades_jornal)
        jc = ManoObra.objects.get(cargo='Jefe de Cuadrilla')
        pu_m = PuMaterial.objects.filter(puma_id=pk)
        pu_mo = PuManoObra.objects.filter(pumo_id=pk)
        tm = pu_m.aggregate(Sum('subtotal'))['subtotal__sum']
        tmo = pu_mo.aggregate(Sum('mo_subtotal'))['mo_subtotal__sum']
        tmo = (tmo + ((jc.salario*8/208) * 0.10))/rm
        total_herramientas = tmo * float(v.herramientas)
        total_seguridad = tmo * float(v.seguridad)
        subtotal = tm + tmo + total_herramientas + total_seguridad
        total = subtotal / float(1 - v.gastos_admin)
        total = total/float(1-v.utilidad)
        total = total*float(v.iva)
        self.valor_mo = round(total * (tmo/subtotal), 2)
        self.valor_material = round(total * (tm/subtotal), 2)

        super().save(*args, **kwargs)
    else:
        super().save(*args, **kwargs)
```

Figura 13 - Método que calcula los valores al grabar.

Fuente: Propria.

Después del llenado de la base de datos con la data necesaria, la parte de cálculos se realizará al momento en que se grabe una instancia u objeto de la tabla de Precios Unitarios. Podemos observar el proceso del flujo de datos en la figura 14. Las tablas intermedias entre materiales y precios unitarios, mano de obra y precios unitarios son necesarias para garantizar que más de un registro de material y mano de obra sea adicionado a la base de datos, sin estas tablas (P.U.Mt y P.U.Mo) solamente se podría grabar una instancia en la tabla de Precios Unitarios.

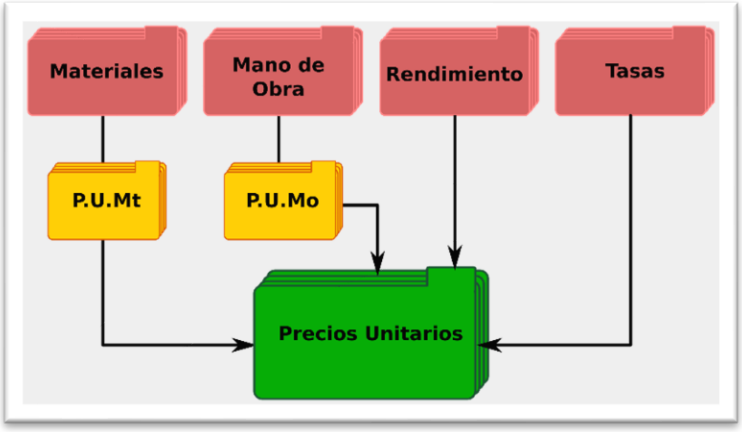


Figura 14 - Modelos y flujo de datos en la base de datos para hacer el análisis del Precio Unitario.
Fuente: Propia.

Para los cálculos del precio unitario, consideramos el subtotal de los materiales (Sub_{mat}), el subtotal de la mano de obra (Sub_{mo}), el valor del rendimiento (Rm) relativo a cuantas unidades se puede realizar por cada período de 8 horas de trabajo, o sea, un jornal de trabajo. Para el caso de la mano de obra, se considera el valor de un supervisor o jefe de cuadrilla además del especialista Electricista y ayudante. Se considera los valores integrales del valor jornal para especialista y ayudante, mientras que se considera para el jefe de cuadrilla un 10% de un jornal, ya que supervisa a más trabajadores en uno solo jornal. Entonces, para calcular el subtotal de la mano de obra (Sub_{mo}) se necesita la cantidad de especialistas (ctd_e), la cantidad de ayudantes (ctd_a) los valores de sus respectivos jornales ($V_{j_{JC}}, V_{j_{esp}}, V_{j_{ayu}}$).

$$Sub_{mo} = \frac{V_{j_{JC}} * 10\% + V_{j_{esp}} * ctd_e + V_{j_{ayu}} * ctd_a}{Rm} \cdots (9)$$

Por lo tanto, el total gastos administrativos sería la suma del subtotal de materiales con el subtotal de la mano de obra dividida entre el número de unidades que se pueda hacer

en un jornal de trabajo. Esto nos brinda el precio de una unidad, pero aún sin gastos administrativos mencionados anteriormente.

$$Total_{Sin\ encargos} = Sub_{mat} + Sub_{mo} \cdots (10)$$

Para hallar el valor del precio unitario, último en análisis, se tiene que agregar los valores porcentuales ($Valor_{her}, Valor_{epp}, Valor_{adm}, Valor_U, Valor_{Con\ encargos}$) de desgaste de herramientas ($\%_{her}$), de los equipos de protección ($\%_{epp}$), ambos repercuten en el valor de la mano de obra (Sub_{mo}), gastos administrativos ($\%_{adm}$), la utilidad deseada ($\%_U$) y finalmente la tasa nominal del IVA (iva_{TN}). Como se muestra en las fórmulas abajo:

$$Valor_{her} = Sub_{mo} * \%_{her} \cdots (11); \qquad Valor_{epp} = Sub_{mo} * \%_{epp} \cdots (12)$$

$$Valor_{adm} = \left(Total_{\text{sin}\ encargos} + Valor_{her} + Valor_{epp}\right) * \%_{adm} \cdots (13)$$

$$Valor_U = \left(Total_{\text{sin}\ encargos} + Valor_{her} + Valor_{epp} + Valor_{adm}\right) * \%_U \cdots (14)$$

$$Valor_{Con\ encargos} = \left(Total_{\text{sin}\ encargos} + Valor_{her} + Valor_{epp} + Valor_{adm} + Valor_U\right) * iva_{TN} \cdots (15)$$

El precio neto es considerado la suma de todos los valores sin el cálculo del IVA, el precio bruto es el precio neto agregando el porcentaje del IVA. En nuestro caso $Valor_U$ sería nuestro precio neto, y el $Valor_{con\ encargos}$, sería nuestro precio bruto.

3.3.3.1 Materiales

La tabla que, en Django son los modelos, son declarados como clases Python. Son objetos que tienen los atributos: nombre, marca, descripción, precio y unidad. Esta

relación es responsable por proporcionar al sistema los datos requeridos de los ítems eléctricos que serán empleados en el análisis del precio unitario que utilice dichos materiales. Toda la base de cálculos de precios depende de este valor, del precio unitario.

```
class Materiales (models.Model):
    UNIDADES_CHOICES = [
        ('PZA', 'PIEZA'),
        ('UND', 'UNIDAD'),
        ('M', 'METRO LINEAR'),
        ('KG', 'KILOGRAMO'),
    ]
    material_nombre = models.CharField('Nombre', max_length=120)
    marca = models.CharField('Marca', max_length=120)
    descripcion = models.CharField('Descripción', max_length=300)
    precio = models.FloatField('Precio')
    unidad = models.CharField('Unidad',
                              max_length=3, choices=UNIDADES_CHOICES, default='PZA')
```

Figura 15 - La clase Python que origina la tabla "Materiales".

Fuente: Propia.

3.3.3.2 Rendimientos

Sin los rendimientos, es imposible hallar los valores unitarios de mano de obra de manera correcta y precisa. Estos valores son fruto de una exhaustiva observación práctica de los "n" tiempos de ejecución (t_{exec}) observados en el proceso de instalaciones eléctricas. Para calcular los rendimientos de una determinada actividad, procedemos de la siguiente manera:

$$t_{Jornal} = 8[h] \cdots (15); t_{muerto} = 50[min] * \frac{1[h]}{60[min]} = 0{,}833[h] \cdots (16)$$

$$t_{jornal_{real}} = t_{jornal} - t_{muerto} \cdots (17) = 8 - 0.833 = t_{jornal_{real}} = 7.167[h]$$

$$Promedio = \frac{\sum t_{exec}[min]}{n} \cdots (18) \therefore \ Promedio\ hora * \frac{1h}{60[min]}$$

$$Unidaddes/Jornal = \frac{Jornal\ Real * 1unidad}{Promedio\ hora} = Rendimiento \cdots (19)$$

El rendimiento, por lo tanto, viene a ser un análisis de los tiempos que los profesionales emplean en determinada actividad o tarea, normalmente registrado en minutos por actividad. Obteniendo un valor promedio de los tiempos en minutos, convertimos el resultado en términos de hora y de esta forma podemos saber en un jornal, cuanto rendirá los profesionales envueltos en estas actividades.

Un análisis de tiempos para el cálculo de rendimientos no viene a ser el alcance del presente proyecto de grado, sin embargo, debe ser una funcionalidad extremadamente valiosa a considerarse en futuras versiones del sistema. Cuanto mejor se logre obtener los rendimientos, más preciso será el análisis de precio unitario, pues su calidad tiene impacto directo en la obtención del precio final.

El módulo de rendimientos de igual manera se declara como una clase Python que se puede observar en la figura abajo.

```python
class Rendimientos(models.Model):
    nombre = models.CharField('Nombre', max_length=120, unique=True)
    descripcion = models.CharField('Descripción', max_length=200)
    unidades_jornal = models.FloatField(default=1.0)

    class Meta():
        ordering = ['nombre']

    def __str__(self):
        return self.nombre + " - " + self.descripcion

    def get_absolute_url(self):
        return reverse('rendimiento_detail', kwargs={'pk': self.pk})
```

Figura 16 - Módulo de Rendimientos.

Fuente: Propia.

3.3.3.3 Mano de Obra

Similarmente que los módulos anteriores, la mano de obra se define en una clase que determina la estructura de datos. La mano de obra y sus cálculos tienen impacto directo en el resultado para un buen análisis de precio unitario.

```python
class ManoObra(models.Model):
    cargo = models.CharField('Cargo', max_length=150, unique=True)
    salario = models.FloatField('Salario')
```

Figura 17 - Módulo Mano de Obra.

Fuente: Propia.

Podemos observar que para los datos de mano de obra solo es necesario dos campos de datos para el alcance de este proyecto de grado técnico.

3.3.4 MÓDULO DE COTIZACIÓN

En el módulo de Cotizaciones no hay muchos cálculos complejos como para el módulo de Precios Unitarios. Su estructura de relacionamientos de datos se ve más simplificada. Como se muestra en el esquema:

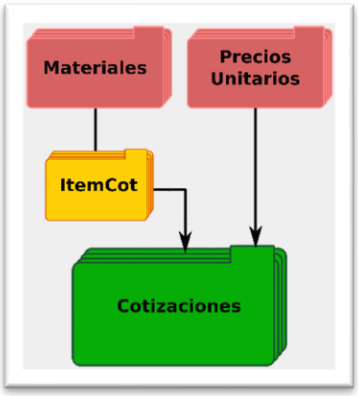


Figura 18 - Modelos y flujo de datos - Cotizaciones.
Fuente: Propia.

Normalmente, cuando no se tiene un módulo de análisis de precios unitarios sería apenas el caso de obtener un set de datos desde la base de datos respecto a los materiales, leer sus valores y multiplicarlos por la cantidad del material. Mayormente se omite la parte del análisis, por lo tanto, no se conoce un valor preciso, es meramente un valor aproximado o deducido.

En el sistema desarrollado se remplaza los datos sin gastos de los materiales, que en los casos ordinariamente se omite la mano de obra utilizando solamente los valores de venta, y retorna como lista de materiales los valores calculados juntamente a los de mano de obra. Recordando que como se está direccionando a una prestación de servicio, la venta de materiales no es la actividad final. Es seguro decir que, con la automatización se viabiliza un precio más justo y aumenta tanto la rentabilidad, como la estabilidad del mismo.

3.3.4.1 Clientes

En el esquema anterior el módulo Clientes fue omitido por solo contar con un campo (nombre) para vincularse a la cotización y no contribuir para al alcance del proyecto.

```
class Clientes(models.Model):
    nombre = models.CharField('Nombre', max_length=200)
```

Figura 19 - Módulo Clientes.

Fuente: Propia.

3.3.5 FUNCIONAMIENTO

Figura 20 - Pagina inicial sin usuario.

Fuente: Propia.

Para operar el sistema, se debe acceder al registro del sistema. Desde la pantalla principal “Cotiza Home”, o vía el enlace “*login*” o marcando el botón al costado superior derecho. El formulario de *login* se carga y luego el usuario debe colocar el nombre de usuario y su contraseña.

Figura 21 - Formulario de Login.

Fuente: Propia.

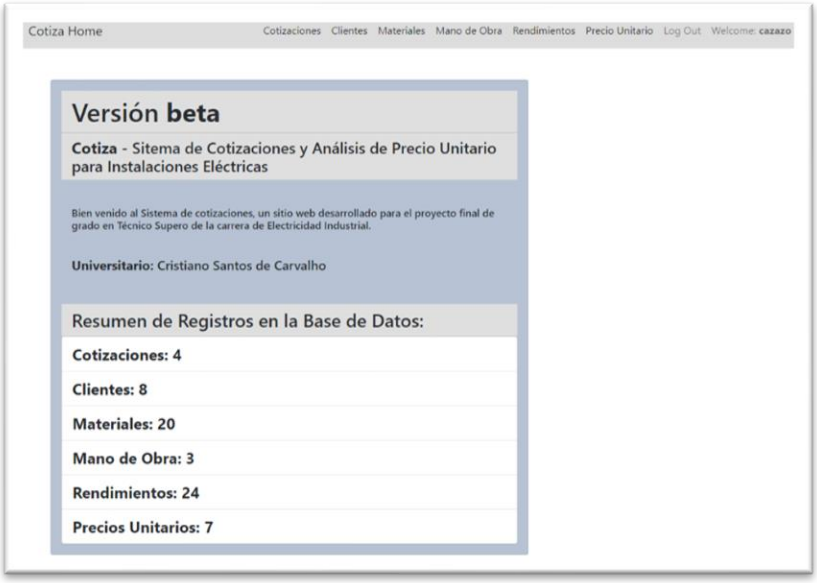


Figura 22 - Vista principal del sistema con Login.

Fuente: Propia.

Los pasos lógicos para describir el funcionamiento, pasan por una secuencia bien definida para llegar a calcular el precio unitario. Vamos a nombrar los pasos secuenciales realizados para la operación del sistema:

1. Análisis de Precios Unitarios
    1.1. Adicionar materiales – en “Materiales”;
    1.2. Adicionar datos de Mano de Obra – en “Mano de Obra”;
    1.3. Ingresar datos de Rendimientos – en “Rendimientos”;
    1.4. Crear nuevo Precio Unitario;
        1.4.1. Escoger el Rendimiento;
        1.4.2. Agregar Materiales y sus cantidades;
        1.4.3. Agregar Mano de Obra y sus cantidades
        1.4.4. Marcar en Guardar;

Para el módulo de Cotizaciones los pasos serán:

2. Cotización;
    2.1. Añadir cliente – en “Clientes”;
    2.2. Crear nueva cotización
        2.2.1. Escoger el cliente;
        2.2.2. Ingresar Materiales y sus cantidades;
        2.2.3. Marcar en Adicionar;

Algunas convenciones importantes de la operación del sistema es que cuando se crea algún registro, automáticamente se direcciona hacia la página de detalles del registro añadido. Hay una consistencia en la apariencia por todo el sistema, en los formularios, las listas y las vistas.

Ejemplos de lista y formulario:

Figura 23 - Lista de Cotizaciones.

Fuente: Propia.

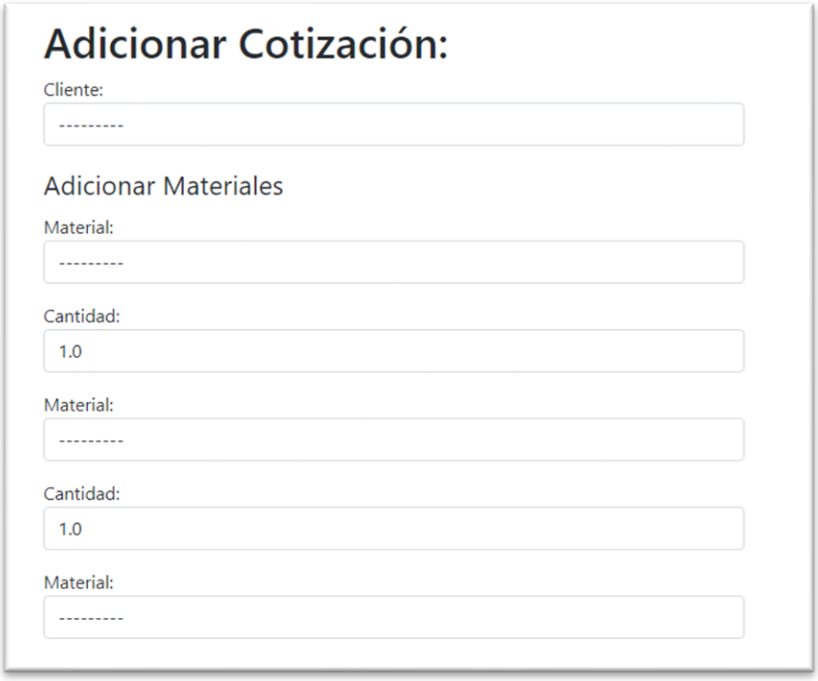


Figura 24 – Formulario Ingreso de Clientes.

Fuente: Propia.

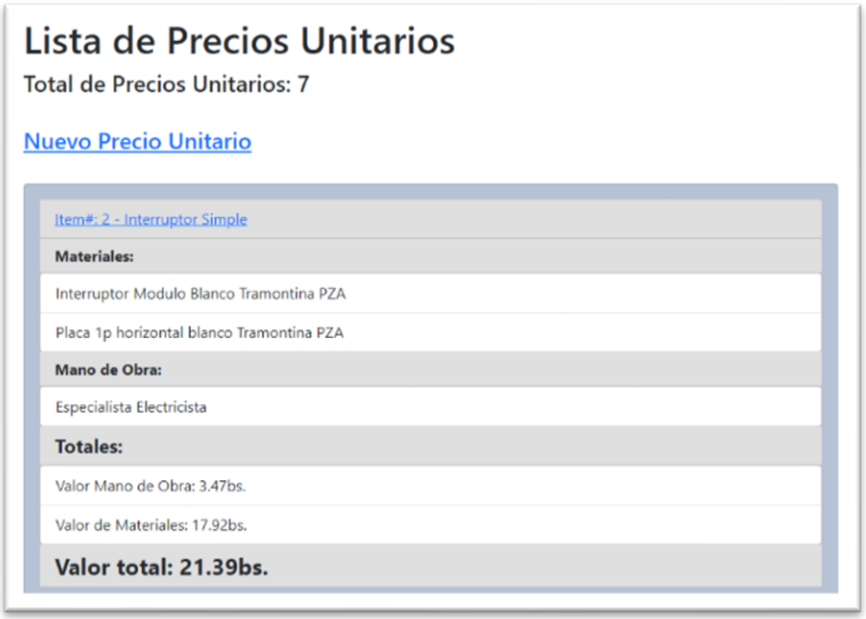


Figura 25 - Lista de Precios Unitarios.

Fuente: Propia

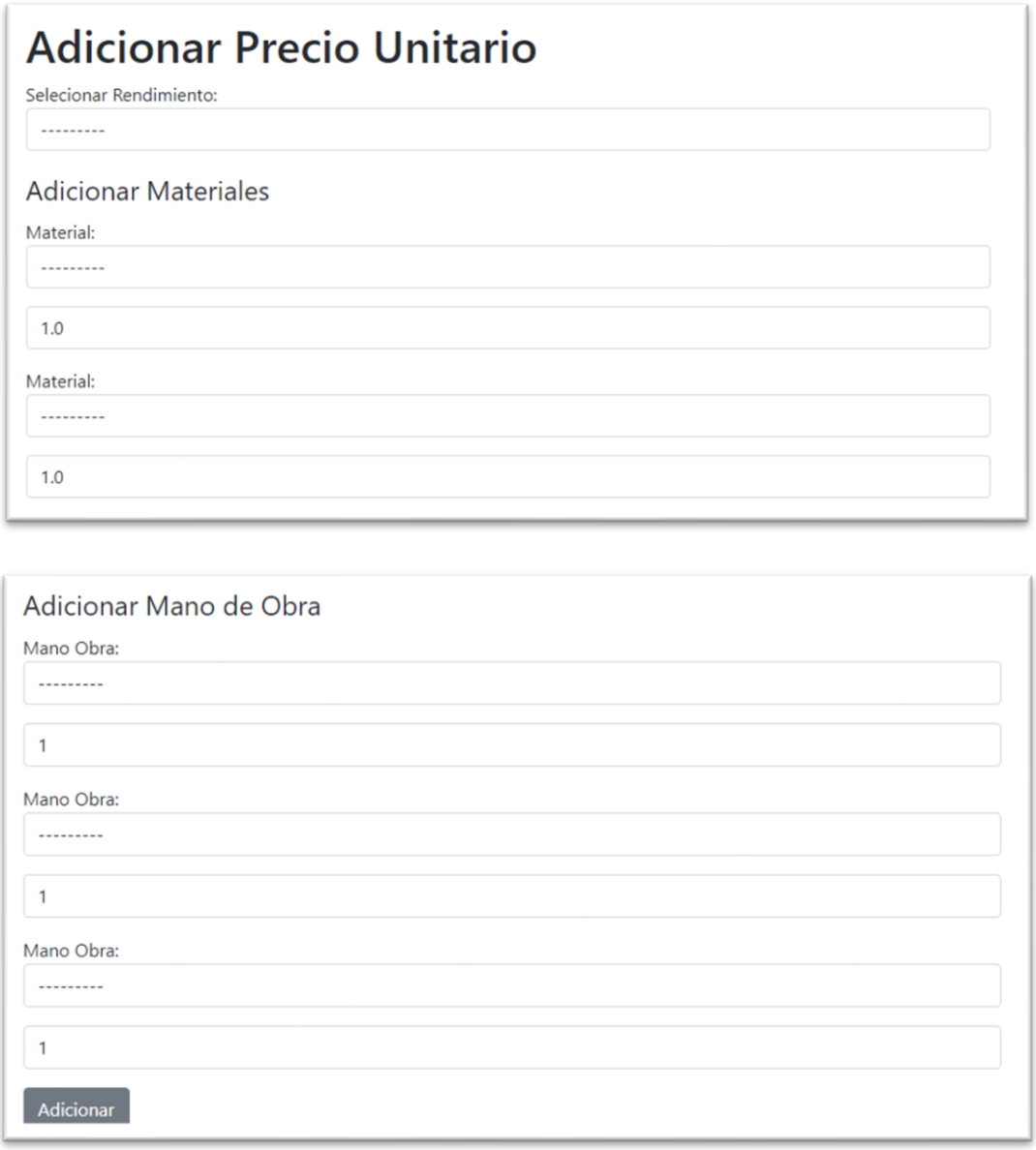


Figura 26 - Formulario de Precio Unitario.

Fuente: Propia

Capítulo IV

Conclusiones y Recomendaciones

4. CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

El objetivo general de desarrollar un sistema para crear cotizaciones y análisis de precios unitarios automatizados, con los debidos gastos y porcentajes, se lograron en su totalidad de manera óptima, consiguiendo llegar a las siguientes conclusiones:

- Se ha alcanzado implementar el presente sistema que fue enteramente diseñado e implementado para la plataforma web;
- En el proceso de investigación, se ha encontrado las tecnologías más recientes utilizadas y con capacidad de expandir modularmente, de fácil mantenimiento y presentando el nivel de estabilidad deseado. Todas las tecnologías fueran utilizadas como complemento una de la otra;
- El sitio web desarrollado está basado en una base de datos relacional entre tablas facilitando el manejo de datos;
- El sistema está separado por capas lógicas, el llamado *backend* que contiene a la base de datos, métodos y reglas de negocio, además el *frontend* que utiliza las plantillas en *html* para mostrar las informaciones por pantalla e ingresar dados por parte del usuario;
- Se comprueba la funcionalidad deseada tanto para hallar los valores de precio unitario, como para generar registros de los datos necesarios, incluso generar una cotización, cumpliendo con todo el alcance propuesto del presente proyecto de grado técnico.

4.2 RECOMENDACIONES

- Se recomienda buscar nuevas soluciones de implementación y/o integración tecnológica de conceptos que se pueda automatizar e incluir al proyecto o en proyectos afines;
- De igual manera, expandir las funcionalidades para poder dar soporte a los usuarios del sistema con cálculos indirectos que puedan resultar en una mejor

prestación de servicio, como por ejemplo crear un portal para cálculos de rebobinados de motores, de corrección de fator de potencia, por nombrar algunos;

- Desarrollar las funcionalidades actuales para mejoramiento de interacción con el sistema, como mejoramiento de interfaz, formularios con lógica más compleja;
- Estimular la consulta tecnológica digital direccionada para la implementación de software en la carrera de Electricidad Industrial.

Capítulo V

Bibliografía

5. BIBLIOGRAFÍA

Deane, M. T. (2021, May 19). *Top 6 Reasons New Businesses Fail*. Investopedia. https://www.investopedia.com/financial-edge/1010/top-6-reasons-new-businesses-fail.aspx#:~:text=According%20to%20the%20U.S.%20Bureau,to%2015%20years%20or%20more.

Heredia Vargas, O. A. (2007). La locura de la administración pública, es cuestión de gestión. Murillo, Bolivia: Heredia Vargas, Oscar Arnaldo

Krajewski, L. J. (2008). Administración de operaciones: procesos y cadenas de valor (8a. ed.). México: Pearson Educación.

Vincent, W. S. (2020, August 12). Django for Beginners: Build websites with Python & Django 3.1. Boston: WelcomeToCode.

Greenfeld, Daniel, Roy, Audrey. (2021, February 1st). Two Scoops of Django 3.x: Best Practices for the Django Web Framework (5a. ed.). Corona: Two Scoops Press.

Django documentation | Django documentation | Django. (n.d.). Djanto Project. Retrieved June 9, 2021, from https://docs.djangoproject.com/en/3.2/

Python.org. (n.d.). 3.9.6 Documentation. Python Project. Retrieved June 9, 2021, from https://docs.python.org/3/

Anexo

GLOSARIO DE TÉRMINOS

- ***app*** – en Django es el equivalente a una funcionalidad del proyecto.
- ***Banco de Datos Relacional*** – banco de datos en que las tablas de datos están relacionadas entre sí para acceso de los registros;
- ***Base de datos*** – estructura lógica donde se almacenan las tablas de datos con sus respectivos campos;
- ***Bootstrap*** – Una aplicación para decorar la presentación de las páginas en *html*;
- ***Campo*** – atributo de una estructura de datos, o sea, atributo de una tabla del banco de datos;
- ***Clase*** – concepto computacional de una abstracción de un objecto en la Programación Orientada a Objeto;
- ***Código de programación*** – texto formal en lenguaje de programación;
- ***CSS*** – hojas de estilo en cascada, utilizadas para formatear páginas en *html*;
- ***Data warehouse*** – colección de datos históricos de una base de datos para cálculos estadísticos y estratégicos;
- ***Django*** – un *web framework* para desarrollo y/o implementación de aplicación para la web.
- ***foreign key*** – un código que representa a una tabla específica de la base de datos que es utilizada para el banco de datos relacional;
- ***Frontend*** – parte o camada conceptual de la programación en que están los archivos e implementaciones necesarias para colectar o mostrar datos por pantalla.

- ***html*** – lenguaje utilizado para crear páginas que son interpretadas por navegadores web;
- ***Lógica de negocio*** – reglas determinadas para restringir el funcionamiento u comportamiento de una determinada aplicación o sistema;
- ***login*** – acceder a un sistema con el fornecimiento de datos de usuario y contraseña registrados en sistema;
- ***Modules (módulos)*** – en Django los *modules* son las definiciones de la base de datos;
- ***open source*** – código abierto al público bajo determinada licencia de utilización;
- ***PostgreSQL*** – Sistema de gestión de banco de datos relacional;
- ***Python*** – lenguaje de programación para desenvolvimiento de software;
- ***stand alone*** – una aplicación que es instalada en solamente una computadora por vez;
- ***templates*** – en Django son plantillas utilizados como base para estructuración de la presentación de datos para el usuario;
- ***versión beta*** – versión no madura de un software en desarrollo;
- ***views*** – camada del Django que se relaciona con los *modules* (módulos) para la entrada y salida de datos;
- ***Visual Studio Code (VSCode)*** – Editor de código de programación utilizado para implementación de software;

- ***web*** – Redes de computadores inter ligada para integración de datos entre servidores;
- ***web framework*** – un conjunto de funcionalidades de un paquete adentro de una determinada aplicación o lenguaje de programación.
- ***web services*** – un servicio ofrecido desde la web para un fin específico;

EJEMPLO DE CÁLCULOS DE RENDIMIENTOS

Tiempo Muerto	
50 min	0.833 h
Jornal Real	
Jornal	7.167 h

Fórmula utilizada para calcular el Rendimiento

$$Unidaddes/Jornal = \frac{Jornal\ Real * 1unidad}{Promedio\ hora} = Rendimiento$$

Interruptor simple, doble y triple:

Item	Objeto	Desc.	Unidad	Tiempo		Mano de Obra		Promedio Min	Promedio Hora	Rendimiento
				Tiempo	U	Desc.	Cant			
1	Interruptor Simple	Montaje y Conexión Interruptor Simple	U	7	min	Electricista	1	7.5	0.1250	57
2	Interruptor Simple	Montaje y Conexión Interruptor Simple	U	5	min	Electricista	1			
3	Interruptor Simple	Montaje y Conexión Interruptor Simple	U	9	min	Electricista	1			
4	Interruptor Simple	Montaje y Conexión Interruptor Simple	U	8	min	Electricista	1			
5	Interruptor Simple	Montaje y Conexión Interruptor Simple	U	9	min	Electricista	1			
6	Interruptor Simple	Montaje y Conexión Interruptor Simple	U	7	min	Electricista	1			
7	Interruptor Doble	Montaje y Conexión Interruptor Doble	U	12	min	Electricista	1	11.6	0.1933	37
8	Interruptor Doble	Montaje y Conexión Interruptor Doble	U	11	min	Electricista	1			
9	Interruptor Doble	Montaje y Conexión Interruptor Doble	U	13	min	Electricista	1			
10	Interruptor Doble	Montaje y Conexión Interruptor Doble	U	12	min	Electricista	1			
11	Interruptor Doble	Montaje y Conexión Interruptor Doble	U	10	min	Electricista	1			
12	Interruptor Triple	Montaje y Conexión Interruptor Triple	U	14	min	Electricista	1	14.0	0.2333	31
13	Interruptor Triple	Montaje y Conexión Interruptor Triple	U	14	min	Electricista	1			
14	Interruptor Triple	Montaje y Conexión Interruptor Triple	U	13	min	Electricista	1			
15	Interruptor Triple	Montaje y Conexión Interruptor Triple	U	13	min	Electricista	1			
16	Interruptor Triple	Montaje y Conexión Interruptor Triple	U	16	min	Electricista	1			

Tomacorriente:

Item	Objeto	Desc.	Unidad	Tiempo		Mano de Obra		Promedio Min	Promedio Hora	Rendimiento
				Tiempo	U	Desc.	Cant			
1	Tomacorriente	Montaje y Conexión Tomacorriente Pol.	U	12	min	Electricista	1	8.86	0.1476	49
2	Tomacorriente	Montaje y Conexión Tomacorriente Pol.	U	8	min	Electricista	1			
3	Tomacorriente	Montaje y Conexión Tomacorriente Pol.	U	10	min	Electricista	1			
4	Tomacorriente	Montaje y Conexión Tomacorriente Pol.	U	7	min	Electricista	1			
5	Tomacorriente	Montaje y Conexión Tomacorriente Pol.	U	10	min	Electricista	1			
6	Tomacorriente	Montaje y Conexión Tomacorriente Pol.	U	6	min	Electricista	1			
7	Tomacorriente	Montaje y Conexión Tomacorriente Pol.	U	9	min	Electricista	1			

Luminaria Led:

Item	Objeto	Desc.	Unidad	Tiempo		Mano de Obra		Promedio Min	Promedio Hora	Rendimiento
				Tiempo	U	Desc.	Cant			
1	Luminaria LED	Montaje y Conexión	U	23	min	Elec. + Ayud.	2	18.40	0.3067	23
2	Luminaria LED	Montaje y Conexión	U	15	min	Elec. + Ayud.	2			
3	Luminaria LED	Montaje y Conexión	U	16	min	Elec. + Ayud.	2			
4	Luminaria LED	Montaje y Conexión	U	18	min	Elec. + Ayud.	2			
5	Luminaria LED	Montaje y Conexión	U	20	min	Elec. + Ayud.	2			

Tabla resumen de Rendimientos:

Desc.	Promedio Min	Promedio Hora	Rendimiento U/J
Interruptor Simple	7.50	0.125	57
Interruptor Doble	11.60	0.193	37
Interruptor Triple	14.00	0.233	31
Conmutador Simple	9.67	0.161	44
Conmutador Doble	13.40	0.223	32
Conmutador Triple	14.20	0.237	30
Conmutador Cuatro Vias	16.40	0.273	26
Tomacorriente	8.86	0.148	49
Montaje Conexión Braker	10.33	0.172	42
Tenido de Manguera 1/2"	4.08	0.068	105
Tenido de Manguera 3/4"	4.39	0.073	98
Tenido de Manguera 1"	4.30	0.072	100
Luces + Empalme 2x14AWG	3.74	0.062	115
Tomacorriente + Empalme	3.27	0.054	132
Luminaria LED	18.40	0.307	23
Cableado + Empalme	3.10	0.052	139
Acometida	4.68	0.078	92
Sensor Magnético	3.92	0.065	110
TV Cable	1.42	0.024	302

DATOS INICIALES DEL BANCO DE DATOS

Materiales - Tabla de datos					
#	nombre	marca	descripción	unidad	precio
1	Cable Flex Noglam AWG 10	Nexans	BWF 70°C - 750V 1 x 6 mm²	M	Bs5.45
2	Cable Flex Noglam AWG 12	Nexans	BWF 70°C - 750V 1 x 4 mm²	M	Bs3.77
3	Cable Flex Noglam AWG 14	Nexans	BWF 70°C - 750V 1 x 2.5 mm²	M	Bs2.37
4	Cable Flex Noglam AWG 8	Nexans	BWF 70°C - 750V 1 x 10 mm²	M	Bs9.38
5	Cinta Aislante - 19m	3M	Color Negro	M	Bs12.00
6	Foco dicroico 5W=50W	Philips	E27 6500k 100-240V	PZA	Bs23.86
7	oco Led dicroico Kit Redondo 5W = 50V	Philips	E27 6500k 100-240V	PZA	Bs43.51
8	Interruptor Térmico 2P 2x16A	CAMSCO	CSC C60K 2x16A 6KA DIN	PZA	Bs28.57
9	Interruptor Térmico 2P 2x20A	CAMSCO	CSC C60K 2x20A 6KA DIN	PZA	Bs28.57
10	Interruptor Térmico 2P 2x32A	CAMSCO	CSC C60K 2x32A 6KA DIN	PZA	Bs28.57
11	Interruptor Térmico 3P 3x32A	CAMSCO	CSC C60K 3x32A 6KA DIN	PZA	Bs42.86
12	Luminaria Farol Exterior	Murano	Luminaria Braquete	PZA	Bs65.00
13	Luminaria Plafón 2 luces	Murano	2 focos	PZA	Bs65.00
14	Módulo Conmutador Cruzado Blanco	Tramontina	Liza 10A 250V	PZA	Bs15.12
15	Módulo Interruptor Blanco	Tramontina	Liza 10A 250V	PZA	Bs6.27
16	Módulo Toma Universal Blanco	Tramontina	Liza 10-15A 250V	PZA	Bs5.41
17	Painel Redondo Led Empotrado	Philips	198x33mm 20W 6500K 220-240V	PZA	Bs153.69
18	Placa 1P Horizontal Blanco	Tramontina	Liza 4x2	PZA	Bs5.66

Mano de Obra - Tabla de datos		
#	Cargo	Salario
1	Jefe de Cuadrilla	Bs4,290.00
2	Especialista Electricista	Bs2,990.00
3	Ayudante	Bs2,290.00

Rendimientos - Tabla de Datos			
#	Nombre	Descripción	Rendimiento U/J
1	Conmutador Cuatro Vias	Suministro, Montaje y Conexión	26
2	Conmutador Doble	Suministro, Montaje y Conexión	32
3	Conmutador Simple	Suministro, Montaje y Conexión	44
4	Conmutador Triple	Suministro, Montaje y Conexión	30
5	Interruptor Doble	Suministro, Montaje y Conexión	37
6	Interruptor Simple	Suministro, Montaje y Conexión	57
7	Interruptor Triple	Suministro, Montaje y Conexión	31
8	Luminaria Dicroico	Suministro, Montaje y Conexión	25
9	Luminaria LED	Suministro, Montaje y Conexión	23
10	Luminaria Plafón	Suministro, Montaje y Conexión	23
11	Montaje Conexión Braker	Suministro, Montaje y Conexión	42
12	Sensor Magnético	Suministro, Montaje y Conexión	110
13	Tenido 2x12 AWG (Tomacorrientes)	Suministro, Montaje y Conexión	132
14	Tenido 2x14 AWG (Iluminación)	Suministro, Montaje y Conexión	115
15	Tenido 2x8 AWG (Acometida)	Suministro, Montaje y Conexión	92
16	Tenido de Manguera 1"	Suministro, Montaje y Conexión	100
17	Tenido de Manguera 1/2"	Suministro, Montaje y Conexión	105
18	Tenido de Manguera 3/4"	Suministro, Montaje y Conexión	98
19	Tomacorriente	Suministro, Montaje y Conexión	49
20	TV Cable	Suministro, Montaje y Conexión	302

CÁLCULOS DE MANO DE OBRA

Tablas de Cálculo de mano de obra:

Semanas en el Año	52	
Semanas por Mes	4.33	
Jornal Diário	8h	
	Caballeros	Damas
Jornal Semana	48h	40h
horas Mensuales	208h	173.33h

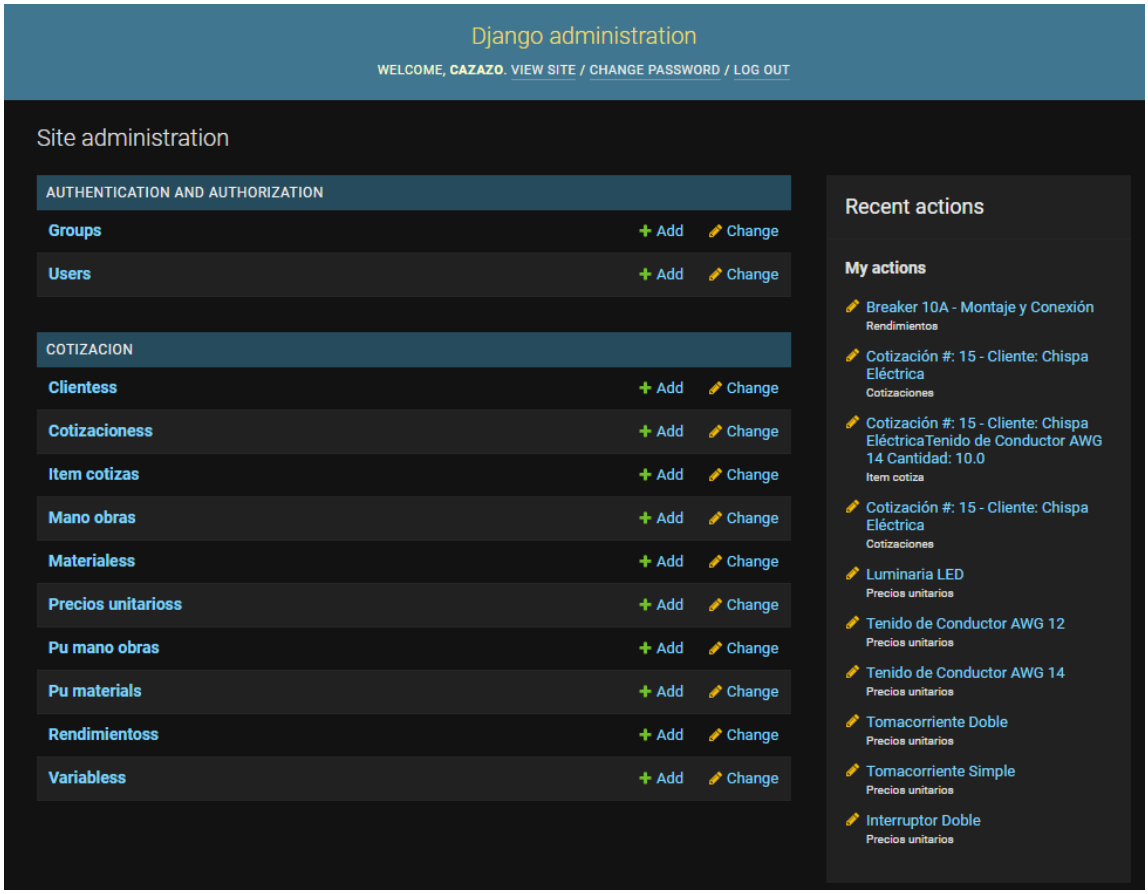
Jefe de Cuadrilla							
Caballero				Dama			
Salário Base	Bs4,290.00	Valor de Hora:	Bs20.63	Salário Base	Bs4,290.00	Valor de Hora:	Bs24.75
Salário Liquido	Bs3,744.74	Jornal(8h)		Salário Liquido	Bs3,744.74	Jornal(8h)	
Descuentos AFP 12,71%	Bs545.26	Bs165.00		Descuentos AFP 12,71%	Bs545.26	Bs198.00	

Especialista Electricista							
Caballero				Dama			
Salário Base	Bs2,990.00	Valor de Hora:	Bs14.38	Salário Base	Bs2,990.00	Valor de Hora:	Bs17.25
Salário Liquido	Bs2,609.97	Jornal(8h)		Salário Liquido	Bs2,609.97	Jornal(8h)	
Descuentos AFP 12,71%	Bs380.03	Bs115.00		Descuentos AFP 12,71%	Bs380.03	Bs138.00	

Ayudante							
Caballero				Dama			
Salário Base	Bs2,290.00	Valor de Hora:	Bs11.01	Salário Base	Bs2,290.00	Valor de Hora:	Bs13.21
Salário Liquido	Bs1,998.94	Jornal(8h)		Salário Liquido	Bs1,998.94	Jornal(8h)	
Descuentos AFP 12,71%	Bs291.06	Bs88.08		Descuentos AFP 12,71%	Bs291.06	Bs105.69	

ÁREA DE ADMINISTRADOR DE BASE DE DATOS

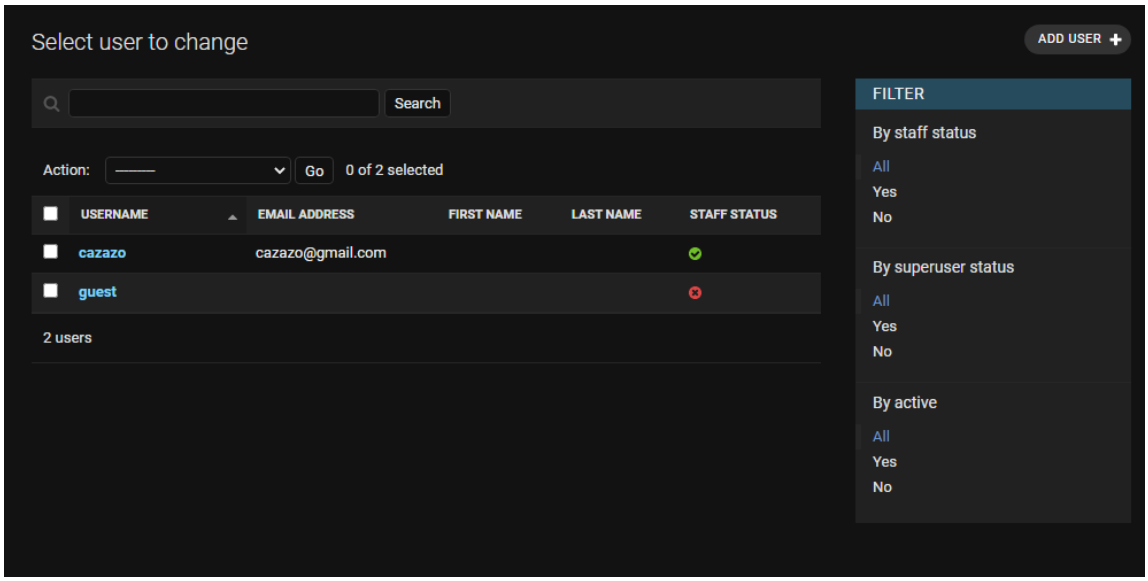
Apenas en esta área es posible utilizar las funcionalidades de actualizar o borrar un registro.



Área del administrador

ADMINISTRACIÓN DE USUARIOS EN EL MODO *ADMIN*:

Solamente en esta parte del *backend* se puede crear, borrar o editar los usuarios que tienen acceso al sistema.



Administración de Usuarios en ambiente Admin

CÓDIGO FUENTE

El código fuente del sistema desarrollado en este proyecto de grado técnico se puede encontrar en su totalidad en el sitio web de Github - https://github.com/cazazo/Cotiza-Precio-Unitario.

MODELS.PY

```
from django.core.exceptions import ValidationError
from django.db import models
from django.db.models import Sum
from django.utils import timezone
from django.urls import reverse
from pandas.core.aggregation import aggregate


def validate_s(value):
    try:
        return round(float(value), 2)
    except:
        raise ValidationError(
            ('%(value)s is not an integer or a float  number'),
            params={'value': value},
        )


class Materiales (models.Model):
    UNIDADES_CHOICES = [
        ('PZA', 'PIEZA'),
        ('UND', 'UNIDAD'),
        ('M', 'METRO LINEAR'),
        ('KG', 'KILOGRAMO'),
    ]
    material_nombre = models.CharField('Nombre', max_length=120)
    marca = models.CharField('Marca', max_length=120)
    descripcion = models.CharField('Descripción', max_length=300)
    precio = models.FloatField('Precio')
    unidad = models.CharField('Unidad',
                              max_length=3, choices=UNIDADES_CHOICES, default='PZA')

    class Meta:
        ordering = ['material_nombre']

    def __str__(self):
        return self.material_nombre+' ' + self.marca + ' ' + self.descripcion + '
'+self.unidad

    def get_absolute_url(self):
        return reverse('material_detail', kwargs={'pk': self.pk})


class Clientes(models.Model):
    nombre = models.CharField('Nombre', max_length=200)

    class Meta:
        ordering = ['nombre']

    def __str__(self):
        return self.nombre
```

```
    def get_absolute_url(self):
        return reverse('cliente_detail', kwargs={'pk': self.pk})


class ManoObra(models.Model):
    cargo = models.CharField('Cargo', max_length=150, unique=True)
    salario = models.FloatField('Salario')

    class Meta:
        ordering = ['cargo']

    def __str__(self):
        return self.cargo

    def get_absolute_url(self):
        return reverse('mo_detail', kwargs={'pk': self.pk})


# Posibilidad de desarrollo --> análisis de rendimientos


class Rendimientos(models.Model):
    nombre = models.CharField('Nombre', max_length=120, unique=True)
    descripcion = models.CharField('Descripción', max_length=200)
    unidades_jornal = models.FloatField(default=1.0)

    class Meta():
        ordering = ['nombre']

    def __str__(self):
        return self.nombre + " - " + self.descripcion

    def get_absolute_url(self):
        return reverse('rendimiento_detail', kwargs={'pk': self.pk})


class Variables(models.Model):
    utilidad = models.FloatField(default=0.15)
    iva = models.FloatField(default=1.1494)
    gastos_admin = models.FloatField(
        default=0.10)
    herramientas = models.FloatField(
        default=0.02)
    seguridad = models.FloatField(
        default=0.04)

    def __str__(self):
        return ("Utilidad: "+str(self.utilidad)+", IVA: "+str(self.iva)
                + ", generales & Administrativos: "+str(self.gastos_admin))


# implementación futura - discriminar los materiales


class PreciosUnitarios(models.Model):
    rendimiento = models.ForeignKey(
        Rendimientos, on_delete=models.CASCADE, related_name='rendimiento')
    mano_obra = models.ManyToManyField(
        ManoObra, through='PuManoObra', related_name='mano_obra')
    material_id = models.ManyToManyField(
        Materiales, through='PuMaterial', related_name='material')
    valor_mo = models.FloatField(
        blank=True, null=True)
    valor_material = models.FloatField(
        blank=True, null=True)

    def __str__(self) -> str:
```

```
            return self.rendimiento.nombre

        def get_total(self):
            return self.valor_mo + self.valor_material

        def get_absolute_url(self):
            return reverse('pu_detail', kwargs={'pk': self.pk})

        def save(self, *args, **kwargs):
            if self.id:
                pk = self.id
                v = Variables.objects.get(pk=2)
                rm = (self.rendimiento.unidades_jornal)
                jc = ManoObra.objects.get(cargo='Jefe de Cuadrilla')
                pu_m = PuMaterial.objects.filter(puma_id=pk)
                pu_mo = PuManoObra.objects.filter(pumo_id=pk)
                tm = pu_m.aggregate(Sum('subtotal'))['subtotal__sum']
                tmo = pu_mo.aggregate(Sum('mo_subtotal'))['mo_subtotal__sum']
                tmo = (tmo + ((jc.salario*8/208) * 0.10))/rm
                total_herramientas = tmo * float(v.herramientas)
                total_seguridad = tmo * float(v.seguridad)
                subtotal = tm + tmo + total_herramientas + total_seguridad
                total = subtotal / float(1 - v.gastos_admin)
                total = total/float(1-v.utilidad)
                total = total*float(v.iva)
                self.valor_mo = round(total * (tmo/subtotal), 2)
                self.valor_material = round(total * (tm/subtotal), 2)

                super().save(*args, **kwargs)
            else:
                super().save(*args, **kwargs)


    class PuManoObra(models.Model):
        manoobra_id = models.ForeignKey(
            ManoObra, on_delete=models.CASCADE, related_name="manoobra_id")
        pumo_id = models.ForeignKey(
            PreciosUnitarios, on_delete=models.CASCADE, related_name='pumo_id', null=True,
blank=True)
        valor_jornal = models.FloatField(
            null=True, blank=True)
        mo_subtotal = models.FloatField(
            null=True, blank=True)
        ctd = models.PositiveIntegerField(default=1)

        def __str__(self) -> str:
            return self.pumo_id.rendimiento.nombre + ", "+self.manoobra_id.cargo

        def save(self, *args, **kwargs):
            vj = round(self.manoobra_id.salario/26, 2)
            self.valor_jornal = vj
            self.mo_subtotal = round(self.ctd*vj, 2)
            super().save(*args, **kwargs)


    class PuMaterial(models.Model):
        material_id = models.ForeignKey(
            Materiales, on_delete=models.CASCADE, related_name='material_id')
        puma_id = models.ForeignKey(
            PreciosUnitarios, on_delete=models.CASCADE, related_name='puma_id', null=True,
blank=True)
        cantidad = models.FloatField(
            default=1.00)
        subtotal = models.FloatField(
            null=True, blank=True)

        class Meta():
```

```
            unique_together = [['material_id', 'puma_id']]

        def __str__(self) -> str:
            return       self.puma_id.rendimiento.nombre       +       ",       Material:
"+self.material_id.material_nombre

        def save(self, *args, **kwargs):
            self.subtotal = round(self.material_id.precio * self.cantidad, 2)
            super().save(*args, **kwargs)


    class Cotizaciones(models.Model):
        cliente_id = models.ForeignKey(Clientes, on_delete=models.CASCADE)
        pu_id = models.ManyToManyField(PreciosUnitarios, through='ItemCotiza')
        valor_mo = models.FloatField(
            blank=True, null=True)
        valor_materiales = models.FloatField(blank=True, null=True)
        creado_en = models.DateTimeField(default=timezone.now)

        class Meta():
            ordering = ['-creado_en']

        def get_valor_total(self):
            return self.valor_mo + self.valor_materiales

        def get_absolute_url(self):
            return reverse('cotizacion_detail', kwargs={'pk': self.pk})

        def __str__(self) -> str:
            return "Cotización #: " + str(self.id) + " - Cliente: " + self.cliente_id.nombre

        def save(self, *args, **kwargs):
            if self.id:
                ic = ItemCotiza.objects.filter(cotizacion_id=self.id)
                tm = ic.aggregate(Sum('valor_material'))['valor_material__sum']
                tmo = ic.aggregate(Sum('valor_mo'))['valor_mo__sum']
                self.valor_materiales = round(tm, 2)
                self.valor_mo = round(tmo, 2)
                super().save(*args, **kwargs)
            else:
                super().save(*args, **kwargs)


    class ItemCotiza(models.Model):
        pu_id = models.ForeignKey(
            PreciosUnitarios, on_delete=models.CASCADE)
        cotizacion_id = models.ForeignKey(
            Cotizaciones, related_name='items', on_delete=models.CASCADE)
        cantidad = models.FloatField(
            default=1.00)
        valor_material = models.FloatField(
            blank=True, null=True)
        valor_mo = models.FloatField(
            blank=True, null=True)

        class Meta():
            unique_together = [['cotizacion_id', 'pu_id']]

        def __str__(self):
            return    str(self.cotizacion_id)+self.pu_id.rendimiento.nombre+"    Cantidad:
"+str(self.cantidad)

        def get_total(self):
            return (self.valor_mo + self.valor_material)

        def save(self, *args, **kwargs):
            self.valor_material = round(
```

```
            self.pu_id.valor_material * self.cantidad, 2)
        self.valor_mo = round(self.pu_id.valor_mo * self.cantidad, 2)
        super().save(*args, **kwargs)
```

VIEWS.PY

```
from django.db.models import Count
from django.http.response import HttpResponseRedirect
from django.shortcuts import get_object_or_404, redirect, render
from django.views.generic import CreateView, DetailView, ListView, TemplateView

from cotizacion.forms import PrecioUnitarioForm
from cotizacion.models import (Clientes, Cotizaciones, ItemCotiza, ManoObra,
                               Materiales, PreciosUnitarios, PuManoObra,
                               PuMaterial, Rendimientos)

from .forms import *

# Create your views here.


class Home(TemplateView):
    template_name = 'cotizacion/index.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)

        context['ct'] = Cotizaciones.objects.all().count()
        context['cl'] = Clientes.objects.all().count()
        context['m'] = Materiales.objects.all().count()
        context['mo'] = ManoObra.objects.all().count()
        context['rd'] = Rendimientos.objects.all().count()
        context['pu'] = PreciosUnitarios.objects.all().count()

        return context


class CreateMaterialView(CreateView):
    form_class = MaterialesForm
    model = Materiales
    redirect_field_name = 'cotizacion/materiales_detail.html'


class CreateClientView(CreateView):
    form_class = ClienteForm
    model = Clientes
    redirect_field_name = 'cotizacion/clientes_detail.html'


class CreateManoObraView(CreateView):
    form_class = ManoObraForm
    model = ManoObra
    redirect_field_name = 'cotizacion/manoobra_detail.html'


class CreateRendimientoView(CreateView):
    form_class = RendimientoForm
    model = Rendimientos
    redirect_field_name = 'cotizacion/rendimientos_detail.html'


class MaterialesListView(ListView):
```

```
    model = Materiales
    redirect_field_name = 'cotizacion/materiales_list.html'


class ClientesListView(ListView):
    model = Clientes
    redirect_field_name = 'cotizacion/cliente_list.html'


class ManoObraListView(ListView):
    model = ManoObra
    redirect_field_name = 'cotizacion/manoobra_list.html'


class RendimientoListView(ListView):
    model = Rendimientos
    redirect_field_name = 'cotizacion/rendimiento_list.html'


def PrecioUnitarioListView(request):
    pus_list = PreciosUnitarios.objects.all()
    return render(request, 'cotizacion/pu_list.html', {'pus_list': pus_list})


class CotizacionListView(ListView):
    model = Cotizaciones
    redirect_field_name = 'cotizacion/cotizacion_list.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['cotizacion_list'] = Cotizaciones.objects.all()
        return context


def CotizacionDetailView(request, pk):
    cotiza_list = ItemCotiza.objects.filter(cotizacion_id=pk)
    cot_list = Cotizaciones.objects.filter(id=pk)
    context = {'cotiza_list': cotiza_list, 'cot_list': cot_list}
    return render(request, 'cotizacion/cotiza_list.html',
                  context=context)


class MaterialDetailView(DetailView):
    model = Materiales


def PrecioUnitarioDetailView(request, pk):
    pus_list = PreciosUnitarios.objects.filter(id=pk)
    pu_list = PuMaterial.objects.filter(puma_id=pk)
    mo_list = PuManoObra.objects.filter(pumo_id=pk)
    context = {'pu_list': pu_list, 'mo_list': mo_list, 'pus_list': pus_list}

    return render(request, 'cotizacion/pu_detail.html', context=context)


class ClienteDetailView(DetailView):
    model = Clientes


class ManoObraDetailView(DetailView):
    model = ManoObra


class RendimientoDetailView(DetailView):
    model = Rendimientos
```

```
class PrecioUnitarioFormView(CreateView):
    template_name = 'cotizacion/preciosunitarios_add.html'
    model = PreciosUnitarios
    form_class = PrecioUnitarioForm
    success_url = ''

    def get(self, request, *args, **kwargs):
        self.object = None
        form_class = self.get_form_class()
        form = self.get_form(form_class)
        pumaterial_form = PuMaterialFormSet()
        pumanoobra_form = PuManoobraFormSet()
        return self.render_to_response(
            self.get_context_data(form=form,
                                  pumaterial_form=pumaterial_form,
                                  pumanoobra_form=pumanoobra_form))

    def post(self, request, *args, **kwargs):
        self.object = None
        form_class = self.get_form_class()
        form = self.get_form(form_class)
        pumaterial_form = PuMaterialFormSet(self.request.POST)
        pumanoobra_form = PuManoobraFormSet(self.request.POST)
        if (form.is_valid() and pumaterial_form.is_valid() and
                pumanoobra_form.is_valid()):
            return self.form_valid(form, pumaterial_form, pumanoobra_form,)
        else:
            return self.form_invalid(form, pumaterial_form, pumanoobra_form,)

    def form_valid(self, form, pumaterial_form, pumanoobra_form):

        self.object = form.save()
        pumaterial_form.instance = self.object
        pumaterial_form.save()
        pumanoobra_form.instance = self.object
        pumanoobra_form.save()
        form.save()
        return HttpResponseRedirect(self.get_success_url())

    def form_invalid(self, form, pumaterial_form, pumanoobra_form):

        return self.render_to_response(
            self.get_context_data(form=form,
                                  pumaterial_form=pumaterial_form,
                                  pumanoobra_form=pumanoobra_form))


class CotizacionFormView(CreateView):
    template_name = 'cotizacion/cotizaciones_form.html'
    model = Cotizaciones
    form_class = CotizacionForm
    success_url = ''

    def get(self, request, *args, **kwargs):
        self.object = None
        form_class = self.get_form_class()
        form = self.get_form(form_class)
        puitemcotiza_form = PuItemCotizaFormSet()
        return self.render_to_response(
            self.get_context_data(form=form,
                                  puitemcotiza_form=puitemcotiza_form))

    def post(self, request, *args, **kwargs):
        self.object = None
        form_class = self.get_form_class()
        form = self.get_form(form_class)
        puitemcotiza_form = PuItemCotizaFormSet(self.request.POST)
```

```
            if (form.is_valid() and puitemcotiza_form.is_valid()):
                return self.form_valid(form, puitemcotiza_form)
            else:
                return self.form_invalid(form, puitemcotiza_form)

        def form_valid(self, form, puitemcotiza_form):
            self.object = form.save()
            puitemcotiza_form.instance = self.object
            puitemcotiza_form.save()
            form.save()
            return HttpResponseRedirect(self.get_success_url())

        def form_invalid(self, form, puitemcotiza_form):
            return self.render_to_response(
                self.get_context_data(form=form,
                                      puitemcotiza_form=puitemcotiza_form))
```

FORMS.PY

```
   from django import forms
   from django.forms.models import inlineformset_factory
   from django.db.models import fields
   from spyder import widgets

   from cotizacion.models import *


   class MaterialesForm(forms.ModelForm):
       class Meta():
           model = Materiales
           fields = '__all__'
           labels = {
               'material_nombre': '',
               'marca': '',
               'descripcion': '',
               'precio': '',
               'unidad': '',
           }
           widgets = {
               'material_nombre': forms.TextInput(attrs={'class': 'form-control',
                                                         'placeholder':    'Material    -
Nome'}),
               'marca':  forms.TextInput(attrs={'class':  'form-control',  'placeholder':
'Ingrese la Marca'}),
               'descripcion':       forms.TextInput(attrs={'class':       'form-control',
'placeholder': 'Descripción'}),
               'precio':  forms.TextInput(attrs={'class':  'form-control',  'placeholder':
'Precio del Material'}),
               'unidad': forms.Select(attrs={'class': 'form-control'}),
           }


   class ClienteForm(forms.ModelForm):
       class Meta():
           model = Clientes
           fields = '__all__'
           widgets = {'nombre': forms.TextInput(
               attrs={'class': 'form-control', 'placeholder': 'Ingrese Nombre'}), }


   class ManoObraForm(forms.ModelForm):
       class Meta():
```

```
            model = ManoObra
            fields = ('cargo', 'salario')
            labels = {'cargo': '', 'salario': ''}
            widgets = {
                'cargo':  forms.TextInput(attrs={'class':  'form-control',  'placeholder':
'Cargo'}),
                'salario': forms.TextInput(attrs={'class': 'form-control', 'placeholder':
'Salario'}),
            }


    class RendimientoForm(forms.ModelForm):
        class Meta():
            model = Rendimientos
            fields = '__all__'
            labels = {'nombre': '', 'descripcion': '', 'unidades_jornal': ''}
            widgets = {
                'nombre':  forms.TextInput(attrs={'class':  'form-control',  'placeholder':
'Nombre'}),
                'descripcion':       forms.TextInput(attrs={'class':       'form-control',
'placeholder': 'Descripción'}),
                'unidades_jornal':     forms.TextInput(attrs={'class':     'form-control',
'placeholder': 'Número de Unidades/jornal'}),
            }


    class CotizacionForm(forms.ModelForm):
        class Meta():
            model = Cotizaciones
            fields = ['cliente_id',
                      # 'valor_mo',
                      # 'valor_materiales'
                      ]
            labels = {'cliente_id': 'Cliente',
                      # 'valor_mo': '',
                      # 'valor_materiales': ''
                      }
            widgets = {
                'cliente_id': forms.Select(attrs={'class': 'form-control'}),
                'valor_mo': forms.TextInput(attrs={'class': 'form-control', 'placeholder':
'Precio del Material'}),
                'valor_materiales':    forms.TextInput(attrs={'class':    'form-control',
'placeholder': 'Precio del Material'})}


    class ItemCotizaForm(forms.ModelForm):
        class Meta():
            model = ItemCotiza
            fields = '__all__'


    PuItemCotizaFormSet = inlineformset_factory(
        Cotizaciones, ItemCotiza, fields=(
            'pu_id', 'cotizacion_id', 'cantidad'),
        widgets={
            'pu_id': forms.Select(attrs={'class': 'form-control'}),
            'cotizacion_id': forms.Select(attrs={'class': 'form-control'}),
            'cantidad': forms.TextInput(attrs={'class': 'form-control'})
        },
        labels={
            'pu_id': 'Material',
            'cotizacion_id': 'Cotización',
            'cantidad': 'Cantidad'
        }, can_delete=False, extra=10)


    class VariablesForm(forms.ModelForm):
```

```
        class Meta():
            model = Variables
            fields = '__all__'


    # PRECIO UNITARIO


    class PrecioUnitarioForm(forms.ModelForm):
        class Meta():
            model = PreciosUnitarios
            fields = ('rendimiento',
                      #   'mano_obra',
                      #   'material_id',
                      #   'valor_mo',
                      #   'valor_material'
                      )
            labels = {
                'rendimiento': 'Selecionar Rendimiento',
                'mano_obra': 'Selecionar Mano de Obra',
                'material_id': '',
                'valor_mo': '',
                'valor_material': '',
            }
            widgets = {
                'rendimiento': forms.Select(attrs={'class': 'form-control'}),
                'mano_obra': forms.SelectMultiple(attrs={'class': 'form-control'}),
                'material_id': forms.SelectMultiple(attrs={'class': 'form-control'}),
                'valor_mo': forms.TextInput(attrs={'class': 'form-control', 'placeholder':
'Precio del Material'}),
                'valor_material': forms.TextInput(attrs={'class': 'form-control'}),
            }


    PuMaterialFormSet = inlineformset_factory(
        PreciosUnitarios, PuMaterial, fields=(
            'material_id', 'puma_id', 'cantidad'),
        widgets={
            'material_id': forms.Select(attrs={'class': 'form-control'}),
            'puma_id': forms.Select(attrs={'class': 'form-control'}),
            'cantidad':  forms.TextInput(attrs={'class':  'form-control',  'placeholder':
'Cantidad'})
        },
        labels={
            'material_id': 'Material',
            'puma_id': 'id',
            'cantidad': ''
        }, can_delete=False, extra=10)


    PuManoobraFormSet = inlineformset_factory(
        PreciosUnitarios, PuManoObra, fields=('manoobra_id', 'pumo_id', 'ctd'),
        widgets={
            'manoobra_id': forms.Select(attrs={'class': 'form-control'}),
            'pumo_id': forms.Select(attrs={'class': 'form-control'}),
            'ctd':    forms.TextInput(attrs={'class':    'form-control',    'placeholder':
'Cantidad'})
        },
        labels={
            'manoobra_id': 'Mano Obra',
            'pumo_id': 'id',
            'ctd': ''
        }, can_delete=False)
```

URL.PY

```
"""Cotiza URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/3.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include
from cotizacion.views import *

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', Home.as_view(), name='home'),
    path('accounts/', include('django.contrib.auth.urls')),
    #     materiales urls:
    path('materiales/', MaterialesListView.as_view(), name='materiales_list'),
    path('materiales/new/', CreateMaterialView.as_view(), name='materiales_new'),
    path('material/<int:pk>',
         MaterialDetailView.as_view(), name='material_detail'),

    #     Clientes urls:
    path('clientes/', ClientesListView.as_view(), name='clientes_list'),
    path('clientes/new/', CreateClientView.as_view(), name='clientes_new'),
    path('cliente/<int:pk>',
         ClienteDetailView.as_view(), name='cliente_detail'),

    #     Mano de Obra urls:
    path('manoobra/', ManoObraListView.as_view(), name='manoobra_list'),
    path('manoobra/new/', CreateManoObraView.as_view(), name='mo_new'),
    path('mo/<int:pk>',
         ManoObraDetailView.as_view(), name='mo_detail'),

    #     Rendimiento urls:
    path('rendimientos/', RendimientoListView.as_view(), name='rendimientos_list'),
    path('rendimientos/new/', CreateRendimientoView.as_view(),
         name='rendimientos_new'),
    path('rendimiento/<int:pk>',
         RendimientoDetailView.as_view(), name='rendimiento_detail'),

    #     Cotizaciones urls:
    path('cotizaciones/', CotizacionListView.as_view(), name='cotizacion_list'),
    path('cotizaciones/new/', CotizacionFormView.as_view(),
         name='cotizaciones_new'),
    path('cotizacion/<int:pk>', CotizacionDetailView, name='cotizacion_detail'),

    #     Precios Unitarios urls:
    path('pu/', PrecioUnitarioListView, name='pu_list'),
    path('pu/<int:pk>', PrecioUnitarioDetailView, name='pu_detail'),
    path('pu/new/', PrecioUnitarioFormView.as_view(), name='pu_new'),
]
```

BASE.HTML

Archivo *html* utilizado como la base para todos los otros.

```
   <!DOCTYPE html>
   {% load static %}
   <html lang="en" dir="ltr">

   <head>
       <meta charset="utf-8">
       <title>Cotiza</title>
       <!-- BOOTSTRAP -->
       <!-- Latest compiled and minified CSS -->
       <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
           integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">

       <!-- Optional theme -->
       <link                                                          rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap-theme.min.css"
           integrity="sha384-
6pzBo3FDv/PJ8r2KRkGHifhEocL+1X2rVCTTkUfGk7/0pbek5mMa1upzvWbrUbOZ"
crossorigin="anonymous">


       <!-- CUSTOM CSS STYLES -->
       <!-- <link rel="stylesheet" href="{% static 'css/cotizacion.css' %}"> -->

       <!-- Google Fonts -->
       <link          href="https://fonts.googleapis.com/css?family=Montserrat|Russo+One"
rel="stylesheet">
       <style>
           .card-header {
               background-color: rgb(221, 221, 221);
           }

           .card {
               background-color: rgb(180, 192, 209);
           }

           .card-text {
               background-color: honeydew;
           }
       </style>

   </head>

   <body>
       <nav   class="navbar   navbar-expand-lg   navbar-light"   style="background-color:
#dadada;">
           <div class="container-fluid">
               <a class="navbar-brand" href="{% url 'home' %}"> Cotiza Home </a>
               <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
                   data-bs-target="#navbarSupportedContent"                          aria-
controls="navbarSupportedContent" aria-expanded="false"
                   aria-label="Toggle navigation">
                   <span class="navbar-toggler-icon"></span>
               </button>
               <div class="collapse navbar-collapse" id="navbarSupportedContent">
                   <ul class="navbar-nav ms-auto">
```

```
                        {% if user.is_authenticated %}
                        <li class="nav-item">
                            <a  class="nav-link  active"  aria-current="page"  href="{%  url
'cotizacion_list' %}"> Cotizaciones
                            </a>
                        </li>
                        <li class="nav-item">
                            <a  class="nav-link  active"  aria-current="page"  href="{%  url
'clientes_list' %}"> Clientes </a>
                        </li>
                        <li class="nav-item">
                            <a  class="nav-link  active"  aria-current="page"  href="{%  url
'materiales_list' %}"> Materiales
                            </a>
                        </li>
                        <li class="nav-item">
                            <a  class="nav-link  active"  aria-current="page"  href="{%  url
'manoobra_list' %}"> Mano de Obra
                            </a>
                        </li>
                        <li class="nav-item">
                            <a  class="nav-link  active"  aria-current="page"  href="{%  url
'rendimientos_list' %}">
                                Rendimientos </a>
                        </li>
                        <li class="nav-item">
                            <a  class="nav-link  active"  aria-current="page"  href="{%  url
'pu_list' %}"> Precio Unitario </a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="{% url 'logout' %}"> Log Out </a>
                        </li>
                        <li   class="nav-item"><a   class="nav-link"   href="">   Welcome:
<strong>{{user.username}}</strong></a>
                        </li>

                        {% else %}
                        <li class="nav-item">
                            <a class="nav-link" href="{% url 'login' %}">Login <span
                                    class='glyphicon glyphicon-user'></span></a>
                        </li>
                        {% endif %}
                    </ul>
                    <!-- <form class="d-flex">
                        <input class="form-control me-2" type="search" placeholder="Search"
aria-label="Search">
                        <button              class="btn              btn-outline-success"
type="submit">Search</button>
                    </form> -->
                </div>
            </div>
        </nav>
        <div class="container container-fluid">
            <div class="row">
                <div class="col-md-8">
                    <div class="cotiza-contents">
                        {% block content %}
                        <br>
                        {% endblock %}
                    </div>
                </div>
            </div>
        </div>
    </body>

    </html>
```

PU_LIST.HTML

```
   <!DOCTYPE html>
   {% load static %}
   <html lang="en" dir="ltr">

   <head>
       <meta charset="utf-8">
       <title>Cotiza</title>
       <!-- BOOTSTRAP -->
       <!-- Latest compiled and minified CSS -->
       <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
           integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">

       <!-- Optional theme -->
       <link                                                          rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap-theme.min.css"
           integrity="sha384-
6pzBo3FDv/PJ8r2KRkGHifhEocL+1X2rVCTTkUfGk7/0pbek5mMa1upzvWbrUbOZ"
crossorigin="anonymous">


       <!-- CUSTOM CSS STYLES -->
       <!-- <link rel="stylesheet" href="{% static 'css/cotizacion.css' %}"> -->

       <!-- Google Fonts -->
       <link          href="https://fonts.googleapis.com/css?family=Montserrat|Russo+One"
rel="stylesheet">
       <style>
           .card-header {
               background-color: rgb(221, 221, 221);
           }

           .card {
               background-color: rgb(180, 192, 209);
           }

           .card-text {
               background-color: honeydew;
           }
       </style>

   </head>

   <body>
       <nav   class="navbar   navbar-expand-lg   navbar-light"   style="background-color:
#dadada;">
           <div class="container-fluid">
               <a class="navbar-brand" href="{% url 'home' %}"> Cotiza Home </a>
               <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
                   data-bs-target="#navbarSupportedContent"                          aria-
controls="navbarSupportedContent" aria-expanded="false"
                   aria-label="Toggle navigation">
                   <span class="navbar-toggler-icon"></span>
               </button>
               <div class="collapse navbar-collapse" id="navbarSupportedContent">
                   <ul class="navbar-nav ms-auto">
                       {% if user.is_authenticated %}
                       <li class="nav-item">
                           <a  class="nav-link  active"  aria-current="page"  href="{%  url
'cotizacion_list' %}"> Cotizaciones
                           </a>
```

```
                    </li>
                    <li class="nav-item">
                        <a  class="nav-link  active"  aria-current="page"  href="{%  url
'clientes_list' %}"> Clientes </a>
                    </li>
                    <li class="nav-item">
                        <a  class="nav-link  active"  aria-current="page"  href="{%  url
'materiales_list' %}"> Materiales
                        </a>
                    </li>
                    <li class="nav-item">
                        <a  class="nav-link  active"  aria-current="page"  href="{%  url
'manoobra_list' %}"> Mano de Obra
                        </a>
                    </li>
                    <li class="nav-item">
                        <a  class="nav-link  active"  aria-current="page"  href="{%  url
'rendimientos_list' %}">
                            Rendimientos </a>
                    </li>
                    <li class="nav-item">
                        <a  class="nav-link  active"  aria-current="page"  href="{%  url
'pu_list' %}"> Precio Unitario </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'logout' %}"> Log Out </a>
                    </li>
                    <li   class="nav-item"><a   class="nav-link"   href="">   Welcome:
<strong>{{user.username}}</strong></a>
                    </li>

                    {% else %}
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'login' %}">Login <span
                                class='glyphicon glyphicon-user'></span></a>
                    </li>
                    {% endif %}
                </ul>
                <!-- <form class="d-flex">
                    <input class="form-control me-2" type="search" placeholder="Search"
aria-label="Search">
                    <button               class="btn               btn-outline-success"
type="submit">Search</button>
                </form> -->
            </div>
        </div>
    </nav>
    <div class="container container-fluid">
        <div class="row">
            <div class="col-md-8">
                <div class="cotiza-contents">
                    {% block content %}
                    <br>
                    {% endblock %}
                </div>
            </div>
        </div>
    </div>
</body>

</html>
```

pu_detail.html

```
<!DOCTYPE html>
{% load static %}
<html lang="en" dir="ltr">

<head>
    <meta charset="utf-8">
    <title>Cotiza</title>
    <!-- BOOTSTRAP -->
    <!-- Latest compiled and minified CSS -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
        integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">

    <!-- Optional theme -->
    <link                                                    rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap-theme.min.css"
        integrity="sha384-
6pzBo3FDv/PJ8r2KRkGHifhEocL+1X2rVCTTkUfGk7/0pbek5mMa1upzvWbrUbOZ"
crossorigin="anonymous">


    <!-- CUSTOM CSS STYLES -->
    <!-- <link rel="stylesheet" href="{% static 'css/cotizacion.css' %}"> -->

    <!-- Google Fonts -->
    <link          href="https://fonts.googleapis.com/css?family=Montserrat|Russo+One"
rel="stylesheet">
    <style>
        .card-header {
            background-color: rgb(221, 221, 221);
        }

        .card {
            background-color: rgb(180, 192, 209);
        }

        .card-text {
            background-color: honeydew;
        }
    </style>

</head>

<body>
    <nav   class="navbar   navbar-expand-lg   navbar-light"   style="background-color:
#dadada;">
        <div class="container-fluid">
            <a class="navbar-brand" href="{% url 'home' %}"> Cotiza Home </a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
                data-bs-target="#navbarSupportedContent"                          aria-
controls="navbarSupportedContent" aria-expanded="false"
                aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarSupportedContent">
                <ul class="navbar-nav ms-auto">
                    {% if user.is_authenticated %}
                    <li class="nav-item">
```

```
                        <a class="nav-link active" aria-current="page" href="{% url
'cotizacion_list' %}"> Cotizaciones
                        </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active" aria-current="page" href="{% url
'clientes_list' %}"> Clientes </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active" aria-current="page" href="{% url
'materiales_list' %}"> Materiales
                        </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active" aria-current="page" href="{% url
'manoobra_list' %}"> Mano de Obra
                        </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active" aria-current="page" href="{% url
'rendimientos_list' %}">
                            Rendimientos </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active" aria-current="page" href="{% url
'pu_list' %}"> Precio Unitario </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'logout' %}"> Log Out </a>
                    </li>
                    <li  class="nav-item"><a  class="nav-link"  href="">  Welcome:
<strong>{{user.username}}</strong></a>
                    </li>

                    {% else %}
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'login' %}">Login <span
                                class='glyphicon glyphicon-user'></span></a>
                    </li>
                    {% endif %}
                </ul>
                <!-- <form class="d-flex">
                    <input class="form-control me-2" type="search" placeholder="Search"
aria-label="Search">
                    <button             class="btn             btn-outline-success"
type="submit">Search</button>
                </form> -->
            </div>
        </div>
    </nav>
    <div class="container container-fluid">
        <div class="row">
            <div class="col-md-8">
                <div class="cotiza-contents">
                    {% block content %}
                    <br>
                    {% endblock %}
                </div>
            </div>
        </div>
    </div>
</body>

</html>
```

PRECIOSUNITARIOS_ADD.HTML

```
{% extends 'cotizacion/base.html' %}
{% block content %}
<br><br>

<h1>Adicionar Precio Unitario</h1>
<form action="." method="post">
    {% csrf_token %}
    <div>
        {{ form.as_p }}
    </div>
    <legend>Adicionar Materiales</legend>
    {{ pumaterial_form.management_form }}
    {{ pumaterial_form.non_form_errors }}
    {% for form in pumaterial_form %}
    {{ form.as_p }}
    <div>
        {{ form.description.errors }}
        {{ form.description.label_tag }}
        {{ form.description }}
    </div>
    {% endfor %}
    <legend>Adicionar Mano de Obra</legend>
    {{ pumanoobra_form.management_form }}
    {{ pumanoobra_form.non_form_errors }}
    {% for form in pumanoobra_form %}
    {{ form.as_p }}
    <div>
        {{ form.number.errors }}
        {{ form.number.label_tag }}
        {{ form.number }}
        {{ form.description.errors }}
        {{ form.description.label_tag }}
        {{ form.description }}
    </div>
    {% endfor %}
    <input type="submit" value="Adicionar" class="btn btn-secondary" />
</form>


{% endblock %}
```