Proyecto-----IPC2P2-----WEB-----Grupo #8

202000549 – Luis Daniel Salán Letona
202000363 – Adriana Lucia Ojeda Rivas
202100081 – Javier Andrés Monjes Solórzano
202000416– Jonathan Eduardo Santos Letona
202004769 – Luis Pablo de Jesús López Carrera

Resumen

Se desarrollo un C.R.U.D (Create, Read, Update, Delete) web capaz de ordenar y clasificar la información extraía de un archivo XML e información obtenida por medio de un usuario, ordenamiento que puede variar en función de lo que el usuario desee visualizar. Para lograrlo, se utilizó un Framework utilizando un patrón de diseño del tipo MVC (Modelo Vista Controlador).

El servicio web se implementó utilizando HTML, CSS y JavaScript. La clave para manejar los datos en los objetos creados (POO) fue vital para optimizar la organización de los datos, al abstraer los objetos del mundo real a un entorno informático

Palabras clave

- Python: Lenguaje de programación de alto nivel
- CSS: Lenguaje que permite proporcionar el diseño a documentos electrónicos
- Lista: Tipo de estructura de datos que permite almacenar información dentro de la misma
- XML: lenguaje de marcado extensible que permite almacenar información de manera compartible
- FrameWork: marco de trabajo utilizado para seguir una misma línea de ordenamiento

Abstract

A web C.R.U.D (Create, Read, Update, Delete) was developed, capable of sorting and classifying the information extracted from an XML file and information obtained from a user, sorting that can vary depending on what the user wants to visualize. To achieve this, a Framework was used using a design pattern of the MVC type (Model View Controller).

The web service was implemented using HTML, CSS and JavaScript. The key to manage the data in the created objects (POO) was vital to optimize the organization of the data, by abstracting the objects from the real world to a computer environment.

Keywords

- *Python: High level programming language*
- *CSS: Language that allows to provide the layout to electronic documents.*
- *List: Type of data structure that allows information to be stored within it.*
- *XML: extensible markup language that allows information to be stored in a sharable manner*
- *FrameWork: used to follow the same line of arrangement.*

Introducción

Python es un lenguaje de programación de alto nivel interpretado que se ha vuelto popular en los últimos años. Se caracteriza por su legibilidad, facilidad de aprendizaje y versatilidad, lo que lo hace adecuado para una amplia gama de aplicaciones.

Python en esta ocasión se utilizó para la gestión de los modelos de parte un nuestro framework, en términos un poco más sencillo Python se encarga de mantener funcionando el servidor que permite el funcionamiento de nuestra aplicación Web

Desarrollo del tema

El desarrollo de nuestro proyecto se llevó a cabo de manera estructurada y planificada, centrándonos en la implementación de un C.R.U.D. web eficiente y fácil de usar. Para lograr este objetivo, seleccionamos el framework Django, una elección que resultó fundamental para el éxito de nuestro proyecto.

Django, un framework de código abierto, demostró ser la opción ideal para el desarrollo de nuestra aplicación web. Su arquitectura basada en el patrón de diseño MVC (Modelo Vista Controlador) nos permitió organizar nuestro código de manera modular y separar las preocupaciones relacionadas con la presentación, la lógica de negocio y el manejo de datos.

Dentro de Django, aprovechamos funcionalidades como el enrutamiento de URL, que facilitó la navegación dentro de la aplicación, y los modelos de Django, que simplificaron la interacción con la base de datos. Además, la capacidad de Django para gestionar las operaciones C.R.U.D. de manera nativa agilizó el desarrollo y garantizó un código limpio y mantenible.

Etapas del Desarrollo:

El proceso de desarrollo se dividió en varias etapas cruciales. Comenzamos con una fase de planificación donde definimos los requisitos del sistema y diseñamos la arquitectura general de la aplicación. La fase de diseño de la base de datos fue esencial para establecer la estructura de almacenamiento de la información y garantizar la coherencia en los datos manipulados.

La implementación del frontend y backend se realizó de manera simultánea, permitiendo una integración continua y una rápida identificación de posibles problemas. Las pruebas de unidad se llevaron a cabo en cada fase del desarrollo para garantizar la funcionalidad y fiabilidad de cada componente.

Desafíos Encontrados y Soluciones Aplicadas:

Durante el desarrollo, nos enfrentamos a desafíos que exigieron soluciones creativas. La integración eficiente de las tecnologías frontend y backend presentó algunos obstáculos, pero mediante una colaboración estrecha y la adopción de las mejores prácticas de desarrollo web, logramos superar estas dificultades.

Además, la gestión de grandes cantidades de datos provenientes del archivo XML y la interacción con la información del usuario requerían una atención especial. Implementamos algoritmos de optimización y estrategias de manejo de errores para asegurar un rendimiento óptimo y una experiencia de usuario sin contratiempos.

Roles y Colaboración del Equipo:

La colaboración dentro del equipo fue clave para el éxito del proyecto. Se asignaron roles específicos, incluyendo responsabilidades para el frontend y backend. La comunicación constante y la implementación de metodologías ágiles facilitaron la coordinación efectiva, permitiendo adaptaciones rápidas a medida que evolucionaba el proyecto.

Herramientas y Tecnologías Adicionales:

Junto con Django, utilizamos otras herramientas y tecnologías para optimizar el desarrollo. Git, un sistema de control de versiones, facilitó la colaboración y el seguimiento de cambios. La elección de una base de datos específica, como PostgreSQL, fue estratégica para satisfacer los requisitos de almacenamiento y consultas eficientes.

Integración de CSS, HTML y JavaScript:

La implementación del servicio web se realizó mediante la integración efectiva de HTML, CSS y JavaScript. Estas tecnologías se utilizaron para diseñar una interfaz de usuario atractiva y funcional. La combinación de HTML para la estructura, CSS para el diseño y JavaScript para la interactividad proporcionó una experiencia de usuario fluida y receptiva.

Pruebas y Validaciones:

La calidad del software fue un enfoque central, y se realizaron pruebas exhaustivas en todas las etapas del desarrollo. Las pruebas de unidad garantizaron el funcionamiento correcto de cada componente, mientras que las pruebas de integración aseguraron la coherencia y el rendimiento general de la aplicación.

Iteraciones y Mejoras Continuas:

Adoptamos un enfoque iterativo, realizando revisiones basadas en la retroalimentación del equipo y, cuando fue posible, del usuario. Las actualizaciones y mejoras se implementaron de manera continua, garantizando que la aplicación evolucionara para satisfacer las necesidades cambiantes y proporcionar una experiencia óptima.

Estas consideraciones y enfoques durante el desarrollo contribuyeron al éxito de nuestro proyecto, demostrando la eficacia de la elección de tecnologías y prácticas de desarrollo adecuadas.

Conclusiones

Éxito de la Integración Tecnológica:

En conclusión, el proyecto destacó por el éxito en la integración de tecnologías clave, como el framework Django, HTML, CSS, JavaScript y la manipulación eficiente de datos a través de Python y XML. La elección cuidadosa de estas herramientas permitió el desarrollo de un C.R.U.D. web robusto y adaptable, que cumple con los estándares de eficiencia y usabilidad.

Lecciones Aprendidas y Continuidad del Desarrollo:

Las lecciones aprendidas durante el desarrollo, desde la gestión eficaz de datos hasta la importancia de la colaboración dentro del equipo, destacan la naturaleza iterativa y evolutiva del proceso. A medida que miramos hacia el futuro, reconocemos la necesidad de continuar con mejoras y actualizaciones para mantener la aplicación relevante y eficiente en un entorno tecnológico en constante cambio.

Versatilidad y Potencial de Python:

En última instancia, el lenguaje de programación Python se erige como una herramienta versátil y potente, desempeñando un papel esencial en el mantenimiento del servidor y la gestión de modelos dentro de nuestro framework. Su legibilidad y flexibilidad contribuyeron significativamente a la eficacia del desarrollo, subrayando la importancia de elegir tecnologías que no solo aborden los requisitos actuales, sino que también anticipen futuras necesidades.

Referencias bibliográficas

(S/f). Amazon.com. Recuperado el 28 de diciembre de 2023, de https://aws.amazon.com/es/what-is/django/

Django. (s/f). Django Project. Recuperado el 28 de diciembre de 2023, de https://docs.djangoproject.com/en/5.0/intro/tutorial01/

ANEXOS

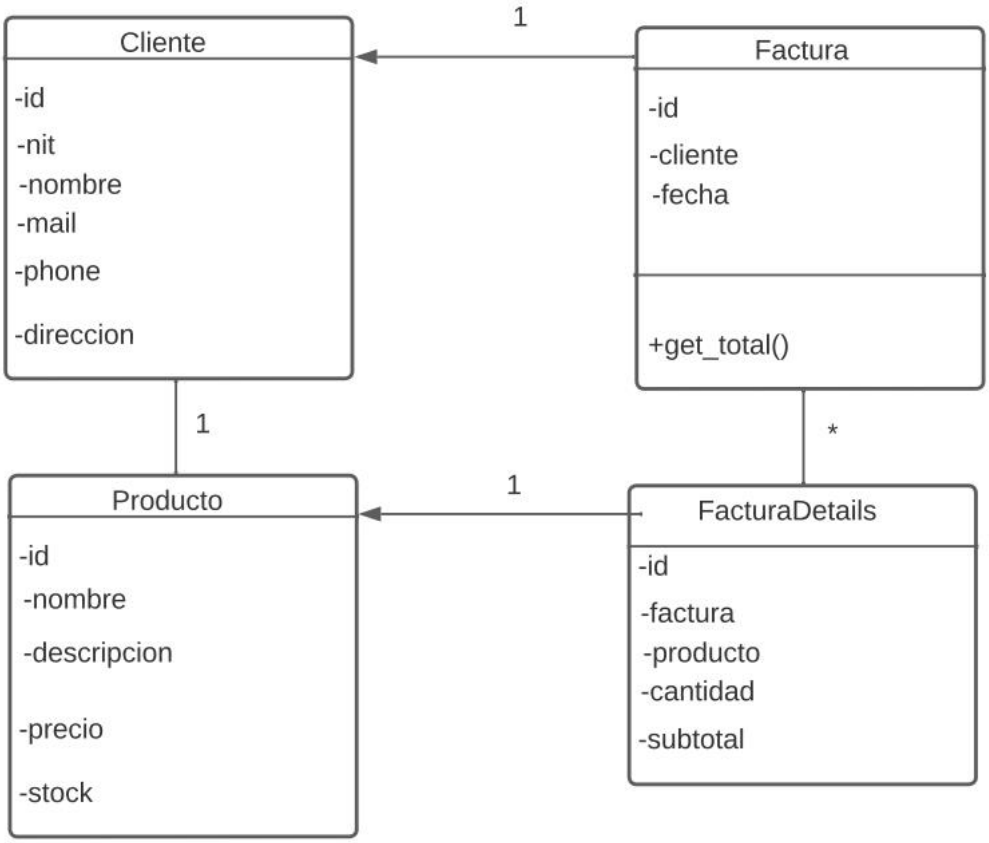
Para poder comprender mejor el proyecto se puede dirigir a los siguiente:

REPOSITORIO

MANUAL USUARIO

MANUAL TECNICO

UML



Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, Escuela de Vacaciones 2023.

Estructura del proyecto:

```
IPC2_Proyecto1Diciembre_-Grupo8
├── Documentación
│   └── Manual Tecnico
│   │   ├── README.md
│   ├── Manual de Usuario
│   │   ├── IMG
│   │   │   ├──imagenes png para el readme.md
│   │   ├── README.md
│   └── Documentación
│   └── Proyecto 2 - IPC2.pdf
├── IMG
│   └── imagenes png para el readme.md inicial
├── project2
│   ├── Backend
│   │   ├──method
│   │   │   ├──__pycache__
│   │   │   |──__init__.py
│   │   │   |──clients.py
│   │   │   |──factura.py
│   │   │   |──product.py
│   │   │   |──readxml.py
│   │   ├──main.py
│   ├── GUI
│   │   ├──__pycache__
│   │   ├──migrations
│   │   │   |__pycache__
│   │   │   |__init__.py
│   │   ├──static
│   │   │   |──Documentacion.pdf
│   │   │   |──estilos.css
│   │   │   |──logo.png
│   │   │   |──logo3.png
│   │   │   |──plantilla.css
│   │   ├──templates
│   │   │   |──publicacion
│   │   │   |   |──clientes.html
│   │   │   |   |──document.html
│   │   │   |   |──factura.html
│   │   │   |   |──index.html
│   │   │   |   |──inicio.html
│   │   │   |   |──productos.html
│   │   ├──__init__.py
│   │   ├──admin.py
│   │   ├──apps.py
│   │   ├──cliente.xml
│   │   ├──factura.xml
│   │   ├──forms.py
│   │   ├──models.py
│   │   ├──stock.xml
│   │   ├──tests.py
│   │   ├──urls.py
│   │   ├──views.py
│   ├── project2
│   │   ├──__pycache__
│   │   ├──__init__.py
│   │   ├──asgi.py
│   │   ├──settings.py
│   │   ├──urls.py
│   │   ├──wsgi.py
│   ├── db.sqlite3
│   ├── manage.py
├── .gitignore
├── README.md
```