Universidad de las Ciencias Informáticas

Facultad 1

Sistema para la Actividad de Control, Influencia y Atención a Personas que Extinguen Sanción en Libertad

Trabajo final presentado en opción al título de

Ingeniero en Ciencias Informáticas

Autor

Hugo Alberto Valencia Zayas

Tutores

MSc. Orlando Grabiel Toledano López

MSc. Juan Manuel Fuentes Rodríguez

Esp. Yanexys González Maceo

La Habana, 2021

Declaración jurada de autoría

Declaro ser autora de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los _____días del mes de _____del año 2021.

Hugo Alberto Valencia Zayas

_____

Firma del autor

MSc. Orlando Grabiel Toledano López

_____

Firma del tutor

MSc. Juan Manuel Fuentes Rodríguez

_____

Firma del tutor

Esp. Yanexys González Maceo

_____

Firma del tutor

Dedicatoria

A mi Madre, quien ha luchado por que sea un hombre de bien y de la cual todos los días aprendo cosas muy buenas, por ser la primera en apoyarme en todas las decisiones que tomo, por tus buenos consejos y tus sabias palabras, este título es para ti, a ti mamá muchísimas gracias.

A mi Padre que hoy no puede verme realizado, a Tata, quien quise mucho y fue como una abuela para mí, a Leandro Alvarez por tanta confianza en mí y por su gran amistad, a elllos que Dios los tenga en la Gloria.

Agradecimientos

Agradezco a mi mama por ser mi todo, por guiarme a ser un buen hombre, mil gracias por todos tus concejos, mil gracias por todos los momentos a tu lado, gracias mama sin ti no hubiese logrado nada.
A mi tutor, hermano y amigo Orlando, gracias por motivarme a estudiar esta carrera, y gracias por todo el tiempo dedicado, gracias por ayudarme a alcanzar logros.
A mi tutor Juan por todo el tiempo dedicado y por sus buenos consejos que me enseñaron mucho.
A mi novia Claudia, gracias por tu incondicional apoyo a mi desarrollo y crecimiento profesional, gracias por la paciencia, gracias por superar la distancia. A Sandra, mi suegra muchas gracias.
A mis amigos de la Uci, Eloy, Iván, Royland, Fabian, Gabriel, Neiser, Dailyn, Marielys, Roxana y Jesús, muchas gracias por hacer de la universidad algo especial para mí.
A Tata, Nancita y Víctor, por su acogida en su seno familiar, por todos esos fines de semanas inolvidables, gracias por ser mi familia. A Tía Nelva, muchas gracias de todo corazón.
A mi familia de las Tunas, Tía Lali y Tío Julio, Luci, Milagrito, Nelson, Hernancitín y Nielita, muchas gracias. A mis hermanos Manuel y Yordano, gracias por su apoyo, por estar siempre.
A mis amigos de Holguín, Danielito, Manoli, Rodolfo, Kassandra, Yisel, Odette, Annie, Gabriel, por todos esos momentos en los que hemos compartido y disfrutado. Gracias a sus familias también.
A Pedri, Bety y Yisel, muchas gracias por siempre estar y por las despedidas cuando tenía que ir para la UCI. A Ernesto, Dianelis y Joaquín por su amistad y su apoyo. A Adriana y su mama Mayi, por ser parte importante y ser parte de mi familia.
A Norlin, Julio y Danel muchas gracias por su amistad. A los muchachos del volley Ale, Elvis, Timón, Wilvian, los del ICPC por los buenos momentos.
A mis compañeros de aula y profesores muchas Gracias
A todos esos buenos amigos que están en todo momento y siempre tienen una muestra de cariño. Muchas Gracias por su amistad.

Resumen

Los tribunales cubanos son el sistema de órganos del Estado al que se le asigna la función de impartir justicia en nombre del pueblo de Cuba. Labor que asumen como servidores públicos, encargados de brindar acceso a la justicia, tutela judicial efectiva y seguridad jurídica a las personas, instituciones, entidades y a la sociedad en general. Los jueces de ejecución, por otra parte, son una figura que atienden, influyen, controlan y ayudan en su reinserción social a las personas egresadas de los centros penitenciarios, ya sea por el beneficio de libertad condicional, o sancionadas a penas que se cumplen de manera condicionada en libertad. En un estudio realizado se identificaron dificultades con la recopilación y el almacenamiento de la información que se gestiona en dos de los procesos. La presente investigación propone un sistema para la gestión de la información asociada a los procesos de la actividad de atención y control de sancionados que extinguen sanción en libertad guiado por la metodología de desarrollo eXtreme Programing. Como lenguajes de programación se utiliza Python, PyCharm como entorno de desarrollo, como lenguaje de modelado UML, Visual Paradigm como herramienta CASE y como servidor de base de datos PostgreSQL. Una vez desarrollado el sistema se efectuaron las pruebas de software a la propuesta de solución, donde las No conformidades fueron resueltas, lo cual valida la calidad de la solución propuesta logrando mayor satisfacción por parte del cliente y garantizando de esta manera el cumplimiento de los objetivos planteados.

Palabras clave

Delito, Django, Ejecución, Juez, Sanción, Sancionado

Abstract

The Cuban courts are the system of State bodies assigned the function of dispensing justice on behalf of the Cuban people. This task they assume as public servants, responsible for providing access to justice, effective judicial protection and legal security to individuals, institutions, entities and society in general. Execution judges, on the other hand, are a figure who attend to, influence, control and assist in the social reintegration of persons released from penitentiary centers, either by the benefit of parole, or punished with sentences that are served conditionally in freedom. In a study conducted, difficulties were identified with the collection and storage of the information managed in two of the processes. The present research proposes a system for the management of information associated with the processes of care and control of the activity of care and control of convicted persons who are released from prison, guided by the eXtreme Programming development methodology. The programming languages used were Python, PyCharm as development environment, UML as modeling language, Visual Paradigm as CASE tool and PostgreSQL as database server. Once the system was developed, software tests were carried out on the proposed solution, where the nonconformities were resolved, which validates the quality of the proposed solution, achieving greater customer satisfaction and thus ensuring compliance with the objectives set.

Keywords

Offense, Django, Enforcement, Judge, Penalty, Sanction, Sanctioned

Índice

Índice de figuras

Índice de tablas

Introducción

Desde los inicios de la humanidad el hombre ha evolucionado en función de sus necesidades, atravesando así por varias etapas hasta llegar a nuestros días, donde el desarrollo de la era tecnológica cada vez toma más auge. Actualmente vivimos en un mundo en el cual sería muy difícil pensar que la tecnología no se encuentre presente en las tareas diarias. Su gran impacto en todos los ámbitos hace cada vez más complejo que se pueda actuar eficientemente prescindiendo de ellas, convirtiéndose en una herramienta esencial para impulsar el desarrollo de una nación (Labañino Urbina et al., 2020).

Cuba no está exenta de esta realidad, por lo que el Gobierno Cubano es consciente de que en los tiempos que corren, para que una sociedad sea más eficaz, eficiente y competitiva es necesario el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC). En nuestro país, el impacto social de la ciencia y la tecnología constituye un tema de actualidad y de particular interés, toda vez que el desarrollo de esta actividad tiene como objetivo principal la sociedad y, por ende, el propio hombre (Batista, 2005).

A del 7mo Congreso del Partido Comunista de Cuba (PCC), fue aprobada la Política Económica y Social vigente en el país. La misma cuenta con lineamientos como el 108 y de manera complementaria otros como 68, 69, 119, 243, 271, los cuales incentivan el desarrollo de la sociedad mediante las TIC. Estos poseen como objetivo elevar la calidad de la información para el ordenamiento territorial en función de lograr mejor eficiencia productiva. Además de prever perturbaciones ambientales, así como la mejora en la velocidad de las comunicaciones en el país y el aumento del suministro de la información de valor para el conocimiento (Presidencia de Cuba, 2019).

Los tribunales cubanos, por mandato del Artículo 147 de la Constitución de la República, son el sistema de órganos del Estado al que se le asigna la función de impartir justicia en nombre del pueblo de Cuba. Como servidores públicos, se encargan de brindar acceso a la justicia, tutela judicial efectiva y seguridad jurídica a las personas, instituciones, entidades y a la sociedad en general. Asegurándoles además amparo legal efectivo a sus derechos y garantías, sobre la base del debido proceso, el cumplimiento de la Constitución y las demás leyes vigentes (Tribunal Supremo Popular, 2020).

Los Tribunales Provinciales Populares (TPP) ejercen su jurisdicción en el territorio de las 15 provincias que comprende la división política–administrativa del país. Existe, además, un

tribunal con categoría provincial en el Municipio Especial Isla de la Juventud. Los TPP tienen su sede, por determinación del Consejo de Gobierno el Tribunal Supremo Popular, en el municipio cabecera de cada provincia. Se integran por su presidente, vicepresidentes, presidentes de salas, y demás jueces profesionales y legos. Los mismos cuentan con unidades administrativas encargadas de asegurar el orden administrativo, en lo que concierne al régimen interior del tribunal y de los tribunales municipales populares de cada provincia, así como otras que garantizan la actividad judicial (Tribunal Supremo Popular, 2019).

Los Tribunales Municipales Populares (TMP) ejercen su jurisdicción en el territorio del municipio donde radican y tienen su sede en la cabecera de estos o en otro lugar dentro del propio territorio, cuando así lo decida el Consejo de Gobierno del Tribunal Supremo Popular. Están integrados por su presidente, presidentes de secciones (cuando estos hayan sido creados para conocer las distintas materias, en consonancia con las necesidades del servicio) y demás jueces profesionales y legos (Tribunal Supremo Popular, 2019).

Los jueces de ejecución, por otra parte, son una figura que atienden, influyen, controlan y ayudan en su reinserción social a las personas egresadas de los centros penitenciarios, ya sea por el beneficio de libertad condicional, o sancionadas a penas que se cumplen de manera condicionada en libertad (Ferro, 2020).

El proceso de control, influencia y atención a personas que extinguen sanción en libertad, es un proceso complejo y se realiza actualmente de forma manual. Todos los procesos asociados al mismo se registran y almacenan utilizando herramientas ofimáticas, que si bien presentan muchas potencialidades también pueden provocar pérdida, duplicidad de información o información errónea. Este proceso incluye un gran cúmulo de datos como todos los datos del sancionado, las visitas que se le realizan ya sea en el domicilio o centro laboral, responsabilidad jurídica y civil, cargos que se le imponen al sancionado, así como el tiempo restante de sanción.

El análisis del funcionamiento del proceso de juez de ejecución permitió identificar las siguientes deficiencias:

- No se dispone de un mecanismo centralizado que gestione el control de los sancionados.

- Existe dificultad para el control de las visitas a los sancionados ya que no se recogen los datos necesarios.

- Existen repetición de datos lo que influye en el proceso y dificulta la obtención de estadísticas certeras de esta actividad.

A causa de estos inconvenientes se obtiene demoras en los procesos de traslados, inserción de radicados sin documentos, obtención del registro de visitas y la obtención de estadísticas confiables. Por lo que se hace necesario la informatización del proceso de control, influencia y atención a personas que extinguen sanción en libertad con el fin de hacer más viable el proceso además de obtener datos con alto nivel de confiabilidad.

A partir de lo planteado se identifica el siguiente **problema a resolver:** ¿Cómo contribuir a la actividad de control, influencia y atención a personas que extinguen sanción en libertad?

El diseño de la investigación permite declarar como **objeto de estudio** el proceso de control judicial a personas que extinguen sanción en libertad, enmarcado en el **campo de acción**: informatización de la actividad de control, influencia y atención a personas que extinguen sanción en libertad en la Sala de Ejecución de los Tribunales Popular Municipales.

Se define como **objetivo general** desarrollar un sistema para el control, influencia y atención a personas que extinguen sanción en libertad en la Sala de Ejecución de los Tribunales Popular Municipales.

Para dar cumplimiento al objetivo general se plantean los siguientes **objetivos específicos:**

1. Establecer el marco teórico para el estudio de la actividad de control, influencia y atención a personas que extinguen sanción en libertad y el desarrollo web.

2. Realizar el Modelado, Requisitos, Análisis y Diseño para la implementación de la propuesta de solución.

3. Implementar una aplicación web para la gestión de la actividad de control, influencia y atención a personas que extinguen sanción en libertad.

4. Validar la solución desarrollada, mediante la aplicación de pruebas de calidad de software que permitan evaluar su valor potencial.

Para el desarrollo de la investigación se emplearon los siguientes **métodos de la investigación científica:**

Métodos teóricos:

- Analítico-Sintético: el uso del mismo permitió descomponer el problema de investigación en varias partes. Con este se pudo profundizar en la actividad del juez de ejecución. De esta forma se pudo determinar los principios fundamentales que debe cumplir la propuesta de solución.

- Histórico-Lógico: se estudió la existencia de sistemas similares, permitiendo así profundizar en las particularidades de cada uno de ellos, con el fin de obtener experiencias propias para la futura implementación.

Métodos empíricos:

- Observación: se utilizó para recopilar información de las necesidades existentes en la actividad de control, influencia y atención a personas que extinguen sanción en libertad.

- Entrevista: se empleó para recoger los criterios de los especialistas que realizan la actividad de juez de ejecución.

Estructura de la tesis

La tesis está compuesta por tres capítulos que responden a los objetivos específicos anteriormente descritos. Cada uno de ellos contiene los elementos necesarios que conforman la solución final, dividiendo el estudio del problema planteado.

En el capítulo 1 se aborda sobre sistemas anteriormente usados para la actividad de juez de ejecución, así como los principales conceptos de esta actividad. Se describen además las tecnologías, metodología, herramientas y el lenguaje de programación utilizado en el desarrollo de la solución.

En el capítulo 2 se exponen los elementos que permiten describir la propuesta de solución, tales como: requisitos para la implementación de un sistema para la actividad del juez de ejecución. Se modelan los aspectos del diseño basados en el diagrama de clases de diseño, también se muestra modelo entidad relación donde se especifican las clases principales con las que interactúa el sistema.

En el capítulo 3 se realiza la implementación la propuesta de solución, obteniendo un sistema de gestión que dota a la actividad de control, influencia y atención a personas que extinguen sanción en libertad. Se describen un conjunto de pruebas realizadas al sistema, una vez que concluye la implementación, asegurando que este cumple con las especificaciones requeridas.

Capítulo 1. Fundamentación Teórica

En el presente capítulo se exponen los principales conceptos a tener en cuenta en el desarrollo de la solución. Se analizan soluciones similares donde se detectan limitaciones. También se describen las diferentes herramientas a utilizar durante el proceso de desarrollo de la solución planteada entre las que se encuentra el IDE, marco de trabajo, gestor de base de datos, así como el uso de patrones de arquitectura

1.1 Principales conceptos

El sistema de juez de ejecución comienza cuando un sancionado obtiene algún privilegio en torno a su sanción, en dependencia del delito cometido. Este se presenta en el TMP donde es atendido por un asistente de ejecución, el cual le toma los datos y lo ingresa al sistema. El sancionado es asignado a un juez de ejecución el cual es el encargado de realizar las visitas al centro de trabajo y zona de residencia. Un sancionado puede trasladarse hacia otro lugar de residencia por motivos personales lo cual debe ser informado. Los sancionados son dados de baja una vez que cumplan con su sanción.

Partiendo de la descripción anterior se obtienen como principales conceptos:

Sancionado: Persona que es penado por un ente regulador a causa de violar una o varias leyes, o cometer algún delito (Alegsa, 2017). En el caso del Juez de Ejecución el sancionado es aquel que obtiene el privilegio de libertad condicional o trabajo correccional sin internamiento.

Juez de Ejecución: Atienden, influyen, controlan y ayudan en su reinserción social a las personas egresadas de los centros penitenciarios, ya sea por el beneficio de libertad condicional, o sancionadas a penas que se cumplen de manera condicionada en libertad (Ferro, 2020).

Delito: Es quebrantamiento de la ley, acción o cosa reprobable penada por la ley. Existen varios tipos como delito común; delito de sangre, que causa lesión corporal grave o muerte; delito político, establecido por los sistemas autoritarios en defensa de su propio régimen (RAE, 2020).

Sanción: Pena que una ley o un reglamento establece para sus infractores. Autorización o aprobación que se da a cualquier acto, uso o costumbre. Acto solemne por el que el jefe del Estado confirma una ley o estatuto. Mal dimanado de una culpa o yerro y que es como su castigo o pena. Estatuto o ley (RAE, 2020).

1.3 Análisis de soluciones similares

En la presente investigación se analizan soluciones existentes en el ámbito nacional e internacional. La primera solución a analizar es un sistema creado usando la herramienta ofimática Access de la suite de Microsoft Office. Solución esta carente de la imagen corporativa de la entidad, además de mal manejo de las relaciones hechas en el modelo relacional. La navegación dentro de esta solución es compleja y todos los datos de los sancionados se pueden modificar sin excepción alguna lo que puede provocar la generación de datos incorrectos, teniendo en cuenta que una vez ingresado un sancionado al sistema hay una serie de datos que no cambian.

De esta primera solución se analizaron dos variantes, una usada por el Tribunal Municipal Popular de Holguín y otra del Tribunal Municipal Popular de Mayarí donde los colores de cada solución son diferentes, no existe homogeneidad en los términos pues unos escriben en abreviatura mientras que en el otro escribir el nombre completo, esto puede causar repetición de datos que si bien puede ser visible en muchos casos en otros no. Los informes estadísticos que genera esta solución son presentados en forma de tablas y no es posible generar un reporte automático de los mismos.

La segunda solución analizada lleva por nombre **SISJE** en su versión 1.0. Es un sistema de gestión desarrollado en el año 2011 por integrantes del departamento de informática del Tribunal Provincial Popular de Santiago de Cuba. SE realizó en lenguaje PHP, HTML, CSS y JavaScript. El mismo fue desplegado para su prueba en 2018 en el Tribunal Provincial Popular de Holguín, donde se detectaron varias deficiencias al sistema como:

- Añadir el delito para tener a vista los priorizados y además lo relativo a la responsabilidad civil si corresponde a persona natural o jurídica y si se encuentra o no pagándola, importante en las exigencias actuales a este tema.

- Incorporar la posibilidad de modificar y actualizar los datos de un sancionado, así como rectificar la sanción o medida que extingue. Dentro de los beneficios solo se hace mención al traslado de los sancionados de un Consejo Popular a otro.

- Se limita el acceso de los asistentes solo a los controlados que atienden, impidiendo el control de esos en caso de que ese asistente se encuentre de vacaciones u otra causa. Esto dificulta el trabajo además de propiciar que se compartan las contraseñas entre asistentes.

- No permite la rotación o cambio de los asistentes que atienden los Consejos Populares, lo cual se hace en un término de 6 meses hasta un año de acuerdo a lo establecido en la Circular 260, lo cual se debe realizar a través del administrador del sistema.

Matrix

Es una solución de gestión de casos legales basada en la web que permite a las empresas del sector legal, como los organismos policiales, los tribunales, las cárceles, los investigadores y los abogados, gestionar documentos, partes interesadas, citas y mucho más. Los usuarios pueden crear informes, organizar sus datos y obtener visibilidad de los flujos de trabajo diarios, los casos o las citas desde un panel de control personalizado (GetApp, 2018b).

Matrix permite a los organismos jurídicos almacenar documentos jurídicos, incluidas las notas, las pruebas, los datos de las partes interesadas y otra información en un expediente electrónico del caso, hacer un seguimiento de las tareas asignadas a los miembros del personal y revisar los resúmenes de los casos judiciales según las necesidades individuales. El portal del cliente ayuda a los clientes a presentar solicitudes, supervisar el progreso del caso, intercambiar notas críticas y revisar los borradores legales. Permite a los usuarios mantener una base de datos de estatutos estatales y federales, crear documentos para hacer un seguimiento de los cargos legales y programar citas en el calendario de Microsoft Outlook mediante la integración con Microsoft Exchange. Además, los defensores pueden ver las notas, crear y asignar tareas, adjuntar multimedia a los archivos del caso y revisar la información legal (GetApp, 2018b).

iJustice

Es una plataforma de gestión de casos legales basada en la web que ayuda a los organismos de aplicación de la ley, los secretarios judiciales, los defensores públicos, los tribunales, los

jueces y los fiscales a gestionar las infracciones de la ley y a automatizar diversas operaciones comerciales. Permite a los usuarios gestionar los procesos de pago, acceder a información sobre eventos o documentos judiciales y programar las salas de audiencia, los jueces, los funcionarios y las partes interesadas asociadas para las próximas audiencias (GetApp, 2019a).

Entre las principales características de iJustice se encuentran la validación de datos, los libros mayores, la conciliación bancaria en línea, las alertas de casos y las notificaciones en tiempo real. Permite a los profesionales del derecho personalizar la interfaz de usuario, compartir información en toda la organización y archivar los documentos almacenados electrónicamente. iJustice permite a los fiscales gestionar las operaciones contables y proteger los datos críticos mediante permisos basados en funciones y capacidades de seguridad multidimensional. El módulo de informes de disposición automatizada (ADR) ayuda a los organismos estatales a producir archivos de disposiciones judiciales con cargos, sentencias y otros detalles del caso extraídos de la base de datos (GetApp, 2019a).

eCourt

Es una solución de gestión de casos basada en la Web para todo tipo de tribunales de justicia que incluyan tribunales de distrito, organismos de jurisdicción del condado y estatal. La solución es ofrecida por *Journal Technologies*, con sede en EE. UU., una entidad resultante de la fusión de tres empresas, *New Dawn Technologies*, *ISD Corporation* y *Sustain Technologies*, todas dedicadas al desarrollo de soluciones para diversas agencias de justicia (GetApp, 2018a).

La solución de administración de casos basada en navegador permite a los usuarios acceder a la información de los casos desde cualquier lugar, actualizar los resultados de las audiencias, administrar las colas de trabajo diario, procesar casos en el tribunal desde el banco y ahorrar tiempo al mantener un repositorio digital centralizado para todos los documentos de los casos. Los usuarios pueden automatizar las tareas repetitivas, tales como los procesos de generación de datos y entrada de datos, para acelerar el flujo de trabajo. La función de acceso permite a los usuarios ver el historial de casos anteriores, agregar documentos a los casos existentes, presentar nuevos casos y realizar pagos electrónicos desde una única interfaz (GetApp, 2018a).

Noble Justice

Es un software de gestión de casos legales que está diseñado para ayudar a los profesionales jurídicos a gestionar diversos procesos de la administración judicial, incluidos la presentación de documentos electrónicos y gestión de libertad vigilada. Los administradores pueden generar informes en tiempo real para realizar un seguimiento del estado de los casos actuales e históricos (GetApp, 2019b).

La plataforma permite a los equipos generar formularios digitales, calcular honorarios y obtener información sobre los expedientes desde un panel centralizado. Noble Justice hace posible que los administradores configuren permisos de acceso basados en roles para los miembros del personal y hagan un seguimiento del historial de los usuarios. Los supervisores también pueden asignar tareas a los empleados en función de los procedimientos judiciales y comunicarse con los miembros del equipo mediante la funcionalidad de chat incorporada (GetApp, 2019b).

Tabla 1: Análisis de soluciones similares

Solución	**Control de visitas**	**Gestión de sancionados**	**Gestión de jueces**	**Gestión de tribunales**	**Idioma español**	**Gratis**
Matrix	-	x	X	x	-	-
iJustice	-	-	-	x	-	x
eCourt	-	-	X	x	-	x
Noble Justice	-	-	-	-	-	x
SISJE	-	x	X	x	x	x
Herramientas Ofimáticas	-	x	-	x	x	x

El estudio de las soluciones analizadas a nivel internacional arroja como resultado que estas solo están disponibles solo para países como los Estados Unidos, estando reguladas bajo la

ley del departamento de tesorería del propio país. Además, el gobierno cubano trabaja para alcanzar la soberanía tecnológica y obtener soluciones ajustadas a la necesidad del país. Tomando en cuenta el analisis de las soluciones nacionales se obtiene que las mismas no cubren las necesidades de la actividad del juez de ejecución. Por todo esto se propone el desarrollo de un sistema para la gestión de la actividad de control y atención de sancionados que extinguen en libertad para el sistema de tribunales cubanos.

1.4 Metodología de desarrollo

Uno de los aspectos fundamentales en la elaboración de un software, es seleccionar las tecnologías que mayores beneficios aporten a la solución informática (Labañino Urbina et al., 2020). Una metodología es un conjunto integrado de técnicas y métodos que permite abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo. Es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito (Maida & Pacienzia, 2015). En el campo del desarrollo de software, existen dos grupos de metodologías, las denominadas tradicionales (formales) y las ágiles (Enríquez Ruiz et al., 2017).

Ambas propuestas tienen sus propias ventajas y desventajas; de cualquier manera, las metodologías de desarrollo nos dicen el *que hacer más no el cómo hacer,* esto significa que la metodología que elijamos, debe ser adaptada al contexto del proyecto, teniendo en cuenta los recursos técnicos y humanos; tiempo de desarrollo y tipo de sistema (Enríquez Ruiz et al., 2017). Actualmente, los ambientes de negocio son muy variables, donde las peticiones del cliente varían con facilidad. Esto hace necesario minimizar el tiempo en las entregas de los productos, por lo que el enfoque ágil prevalece sobre el tradicional, al ser este más dinámico.

Para la elección del enfoque se utilizó como punto de partida el modelo de Barry Boehm y Richard Turner debido a su fiabilidad a la hora de definir el enfoque del proyecto tomando en cuenta sus características, además de ser un método que solo necesita cinco criterios para lograr este objetivo (Toledano & Camps, 2015).

Figura 1: Estrella de Boehm-Turner (Boehm & Turner, 2003)

Parámetros para elección entre una metodología ágil y formal

Para seleccionar la metodología se necesita analizar y medir diversos factores y riesgos asociados al escenario en causa. En el presente trabajo se propone evaluar los siguientes criterios de calidad (Tabla 1) propuesto por Boehm y Turner (Boaventura et al., 2016).

Tabla 2: Criterios para la selección apropiada de la metodología (Boaventura et al., 2016)

Factor	**Discriminadores ágiles**	**Discriminadores formales**
Tamaño	Escalabilidad y conocimiento. Dependencia y escalabilidad limitada por el porcentaje alto de conocimiento tácito. Apropiado para equipos y productos pequeños.	Escalabilidad y conocimiento explícito. Apropiado para productos y equipos grandes. Difícil de mantener en pequeños proyectos.
Criticidad	La simplicidad en la documentación y el diseño dificulta los planes de pruebas. No aconsejado para sistemas con niveles de criticidad altos.	Rigor de requisitos y diseño adecuados para procesos de pruebas, verificación y validación. Difícil de gestionar en proyectos de escasa criticidad.

Dinamismo	“Refactorizar” desde un diseño básico hasta el producto final es un método ideal para entornos dinámicos e innovadores, pero muy caro por el “retrabajo” para entornos estables o conocidos.	En sistemas estables y conocidos, a partir de requisitos completos y diseños detallados permite trazar y seguir un plan completo y “hacerlo bien a la primera”.
Personal	Los métodos de trabajo ágiles requieren una masa crítica de técnicos con niveles de experiencia medios o altos, capaces de comprender y adaptar los métodos y las técnicas empleadas. Consiste en grupos pequeños y trabajando en el mismo local.	Aunque es aconsejable contar con personas expertas en las fases de definición del proyecto, luego pueden ejecutarse con menor masa crítica de expertos. Consiste en grupos grandes y pueden trabajar de forma distribuidas.
Cultura	Más apropiado para culturas de “*empowerment*” responsabilidad y horquilla de decisión y libertad personal.	Más apropiado en culturas en las que las personas se sienten seguras con un marco de tareas y responsabilidades bien definido.

Para el desarrollo del proyecto se cuenta con un equipo constituido por una persona no experta, con conocimientos básicos de las tecnologías a emplear en la solución. El mismo intervendrá en un ambiente dinámico donde las especificaciones y las necesidades del cliente varían. Por otra parte, la criticidad o impacto social del proyecto este se encuentra dentro de la zona de confort donde no existen posibilidades de pérdidas económicas ni de vidas humanas por el uso futuro del sistema, por tanto, se decide utilizar un enfoque ágil.

Existen varias metodologías con enfoque ágil. Analizando las más representativas y utilizadas en el mundo para el proceso de desarrollo de software, a continuación, se exponen alguna de estas.

Proceso Unificado Ágil (AUP):

Desarrollada por *Scott Ambler*, constituye una versión simplificada del Proceso Unificado Racional (RUP, por las siglas de *Rational Unified Process*). Es una metodología que combina procesos propios del concepto unificado tradicional con técnicas ágiles, con el objetivo de mejorar la productividad. Permitiendo así describir de manera simple y fácil de entender cómo desarrollar aplicaciones de software de negocio (Rodríguez Sánchez, 2014). El AUP aplica técnicas ágiles incluyendo:

- Desarrollo Dirigido por Pruebas (*test driven development* - TDD en inglés)
- Modelado ágil
- Gestión de Cambios ágil
- Refactorización de Base de Datos para mejorar la productividad.

Crystal Metodologies

La familia ***Crystal*** es una colección de metodologías ágiles de desarrollo de software que pueden utilizarse para diferentes proyectos de software dependiendo del tamaño, la complejidad, la criticidad y el número de personas involucradas. Fue desarrollada por *Alistair Cockburn* a principios de 1990 mientras trabajando en IBM. Según la filosofía de Cockburn "En la medida en que puedas sustituir la documentación escrita con la interacción cara a cara, se puede mejorar la probabilidad de entregar el sistema". Los métodos Crystal se centran en las personas y en la comunicación entre las personas más que en los procesos para entregar con frecuencia un software que funcione. La familia Crystal incluye una serie de métodos representados por diferentes colores ordenados en opacidad ascendente (Anwer et al., 2017).

SCRUM

Scrum fue desarrollado inicialmente para gestionar y desarrollar productos. Desde principios de los años 90 Scrum se ha usado ampliamente en todo el mundo para (Schwaber & Sutherland, 2017):

- Investigar e identificar mercados viables, tecnologías y capacidades de productos.
- Desarrollar productos y mejoras.
- Liberar productos y mejoras tantas veces como sea posible durante el día.

- Desarrollar y mantener ambientes en la Nube (en línea, seguros, bajo demanda) y otros entornos operacionales para el uso de productos.

- Mantener y renovar productos.

Scrum se ha usado para desarrollar software, hardware, software embebido, redes de funciones interactivas, vehículos autónomos, escuelas, gobiernos, mercadeo, también para gestionar la operación de organizaciones y casi todo lo que usamos en nuestra vida diaria, como individuo y como sociedad (Schwaber & Sutherland, 2017).

eXtreme Programing (XP)

XP es una metodología ágil defendida por *Kent Beck, Ron Jeffries, Ward Cunnighan* entre otros. Se compone de un conjunto de valores y prácticas que están en camino desde mediados de los 90s. Muchos equipos y empresas buscan entender el desarrollo del software desde otra perspectiva. Desde su implementación provocó un cambio en la manera de gestionar proyectos ya que se basa en la perspectiva de la gestión técnica utilizando cada vez menos las prácticas de ingeniería (Villa, 2019).

La programación extrema es una metodología ágil y ligera de desarrollo de software que tiene la capacidad de responder rápidamente a requisitos cambiantes (Sohaib et al., 2018). Además sugiere conseguir diseños simples con el objetivo de hacer todo lo menos complicado para el usuario, haciéndolo entendible y aplicable. Es una metodología ágil debido a que permite priorizar y adoptar control de procesos ya sea de manera empírica o de manera definida (Peñaherrera, 2021). La misma se centra en potenciar las relaciones interpersonales del equipo de desarrollo como clave del éxito mediante el trabajo en equipo, el aprendizaje continuo y el buen clima de trabajo (Villa, 2019).

Esta metodología, trata de realizar ciclos de desarrollo cortos llamados iteraciones, con entregables funcionales al finalizar cada ciclo. En cada iteración se realiza un ciclo completo de análisis, diseño, desarrollo y pruebas, pero utilizando un conjunto de reglas y prácticas que la caracterizan (Alba, 2018). La misma propone técnicas como diseño simple, pruebas intensivas, refactorización y programación en pareja (Pilataxi Alba, 2018).

Ciclo de vida de la Metodología XP

Exploración: Esta fase determina donde se gana confianza las herramientas que se utilizarán, los miembros del ED (Equipo de desarrollo) se conocen y aprenden a confiar unos en otros. Durante esta fase los programadores deben estimar cada tarea de programación en la que hayan sido divididas las historias de usuario, mientras los clientes van escribiendo las historias de usuarios (Pilataxi Alba, 2018).

Planificación: El objetivo de la fase de planificación es que el cliente y el ED lleguen a una serie de acuerdos y se elabore el plan de entrega. En esta fase el cliente asigna prioridad a las historias de usuario, se estima el costo total del sistema, el alcance de las entregas y para qué fecha culminará (Pilataxi Alba, 2018).

Iteraciones: Esta fase incluye varias iteraciones antes de que se libere la primera versión del software. El plan de entrega se divide en diferentes planes de iteración. Cada iteración puede tener de una a cuatro semanas de duración. De la primera iteración se obtendrá la arquitectura del futuro sistema (Pilataxi Alba, 2018).

Producción: Requiere de pruebas y chequeos de funcionamiento y rendimiento del sistema antes de ser liberado a los clientes. Durante esta fase las iteraciones deben dinamizarse, duran entre una y tres semanas. Las nuevas ideas y sugerencias se posponen y se documentan para ser implementadas en la fase de mantenimiento (Pilataxi Alba, 2018).

Mantenimiento: En esta fase, después de que el sistema pasó la fase de producción, la velocidad de desarrollo tiende a disminuir. Esta fase requiere de un cambio en el equipo desarrollo y en su estructura (Pilataxi Alba, 2018).

Muerte: En esta fase el cliente debe quedar satisfecho con el trabajo realizado y no escribir ninguna otra historia de usuario. Se procede a escribir la documentación del sistema si no va a haber más cambios en la arquitectura, ni en el código (Pilataxi Alba, 2018).

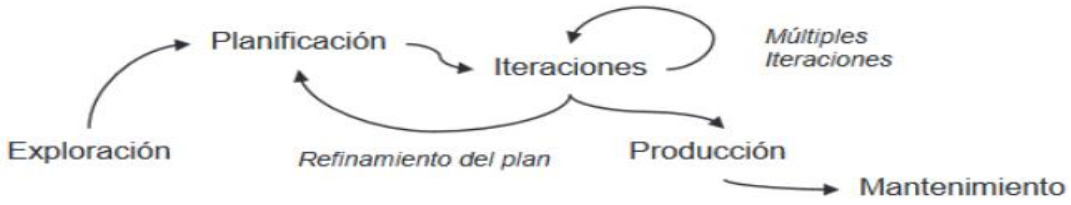

Figura 2: Ciclo de vida de un proyecto basado en XP (Pilataxi Alba, 2018)

Para la presente propuesta de solución se selecciona la metodología XP, esto debido a su tasa de errores muy pequeña. Además, fomenta la comunicación entre el desarrollador y el cliente. Posee también una alta eficiencia en el proceso de prueba y planificación, lo que mejora la experiencia de trabajo de este proyecto que solo cuenta con un desarrollador.

1.5 Descripción de las herramientas y tecnologías para el desarrollo de la propuesta de solución

Selección del frameworks de desarrollo web

Un framework web es un marco de trabajo de software que se desarrolla para simplificar el proceso de construcción de un sitio web. Es la forma estándar de construir y desplegar aplicaciones web en Internet. El objetivo principal de los frameworks para el desarrollo web es automatizar las actividades comunes realizadas durante la fase de desarrollo. Estos frameworks vienen con capacidades de plantillas que permiten presentar la información en un navegador, bibliotecas para el acceso a bases de datos, gestión de sesiones y capacidades de reutilización de código (GMI Blogger, 2021).

A diferencia de otras herramientas de desarrollo, los frameworks vienen con una base de código predefinida y directrices para facilitar el desarrollo web. De este modo, se acelera el proceso de desarrollo y se reduce el tiempo de comercialización (GMI Blogger, 2021).

Estas son las 4 principales ventajas de utilizar frameworks web:

1. Fácil depuración y mantenimiento: La mayoría de los lenguajes de programación no dan importancia a la legibilidad y el mantenimiento del código. Pero muchos marcos de desarrollo de sitios web populares lo hacen. Los frameworks se recomiendan para el desarrollo web a medida porque tienen mucho que ver con la facilidad de depuración y soporte. Hay una comunidad de desarrolladores con cada marco, por lo que puede estar seguro de respuestas rápidas a cualquier problema (GMI Blogger, 2021).

2. Reducir la longitud del código: Si se utilizan *frameworks*, no es necesario escribir largas líneas de código para añadir funcionalidades estándar a un sitio web. Los frameworks vienen con características de generación de código para promover la simplicidad y concisión, reduciendo el tiempo y el esfuerzo requerido por los desarrolladores. Además, los frameworks proporcionan herramientas y funciones que ayudan a los desarrolladores a automatizar tareas

habituales como la autenticación, la asignación de URL, el almacenamiento en caché, entre otros (GMI Blogger, 2021).

3. Seguridad mejorada: Los frameworks proporcionan características y mecanismos de seguridad integrados que ayudan a los desarrolladores a proteger los sitios web de las amenazas de seguridad presentes y futuras. Mediante el uso de frameworks, los programadores pueden proteger los sitios web de diversos ciberataques como la manipulación de datos, DDoS, inyecciones SQL, etc. Además, se pueden utilizar marcos web de código abierto para crear especificaciones de seguridad personalizadas para los sitios web (GMI Blogger, 2021).

4. Mejora la eficiencia y la reutilización del código: Los frameworks web proporcionan a los desarrolladores un entorno de codificación rápido, sensible y eficiente. Además, los frameworks vienen con características avanzadas como la recarga en caliente y la recarga en vivo, lo que lleva a ciclos de desarrollo más rápidos. Además, al utilizar marcos web, los desarrolladores no necesitan escribir líneas de código complejas o múltiples. En su lugar, pueden utilizar la base de código predefinida para realizar modificaciones sencillas y llevar a cabo un arranque fácil (GMI Blogger, 2021).

Symfony

Es un entorno de trabajo estandarizado que se utiliza para el desarrollo de aplicaciones web y es de los más utilizados en el entorno de desarrolladores de apps. En otras palabras, es una herramienta para desarrolladores para crear aplicaciones en PHP. La empresa creadora de este framework fue *SensioLabs* y es utilizada actualmente por miles de empresas de desarrollo web en todo el mundo. Una de las principales ventajas de Symfony es que posee una licencia MIT, o lo que es lo mismo que una licencia de software libre permisivo, que se puede utilizar dentro del software del propietario y al no poseer *Copywright* permite su modificación. La primera versión de Symfony se creó en octubre de 2005 y una de sus curiosidades es que Yahoo! lo eligió como su framework PHP de desarrollo, con el que construyó Yahoo! *Bookmarks*. Para poder trabajar con esta herramienta es necesario disponer de un servidor web, una consola de comandos del sistema operativo y cualquier versión de PHP5 o PHP7 si se quiere utilizar las últimas versiones. El framework de desarrollo de aplicaciones tiene una gran aceptación y popularidad entre programadores de PHP en Europa, lo que hace que tenga una amplia comunidad que ofrece formación, consultorías y desarrollo de proyectos (Quality Devs, 2019).

Laravel

Laravel es una herramienta de código abierto, desarrollada con todas las opciones disponibles para elaborar tus proyectos web sin perder el tiempo creando todo desde cero. Requiere menos código que escribir y tienes un apartado de bibliotecas diseñado para tareas sencillas y comunes. Laravel tiene muchas funcionalidades, con las cuales puedes contar y que te harán preferir esta herramienta por encima de otras. Laravel usa muchas herramientas o utilidades. con distintos fines (Vera, 2021).

Características de *Laravel*:

- Su motor de plantilla, llamado Blade, da numerosas posibilidades para hacer unas páginas visualmente muy potentes y eficaces, capaz de utilizar sus propias variables y reutilizarlas (Vera, 2021).

- Su arquitectura es conocida como MVC (Modelo-Vista-Controlador) que da muchas facilidades para relacionar de manera clara y sencilla todas las partes de una aplicación. Esta arquitectura es muy usada en el mundo del software, otros *framework* pueden distintos de Laravel pueden resultar muy similares gracias a compartir la misma arquitectura MVC (Vera, 2021).

- *Eloquent ORM*, es muy intuitivo para escribir consultas en PHP sobre objetos. Otros *framework* cuenta con Doctrine por ejemplo, otro tipo de ORM que quizás te podría sonar más que el que usa Laravel (Vera, 2021).

- En seguridad, ofrece un nivel bastante fuerte con mecanismos de hash y salt para encriptar por medio de librerías como ***BCrypt***, que también lo usa por ejemplo ***Zend Framework*** (Vera, 2021).

Web2py

Web2py se define como un framework de desarrollo web de código abierto para el desarrollo ágil que implica aplicaciones web con bases de datos. Está escrito en Python y programable. Es un completo *framework* y se compone de todos los componentes necesarios un desarrollador necesita para construir aplicaciones web totalmente funcionales (Di Pierro, 2014).

Sigue el patrón Modelo-Vista-Controlador de ejecutar aplicaciones web a diferencia de los modelos tradicionales. Tiene la capacidad de ejecutar las tareas en intervalos programados después de la realización de ciertas acciones. Múltiples aplicaciones pueden estar alojados en la misma instancia de web2py (Di Pierro, 2014).

Django

Django es un *framework* web de alto nivel que permite el desarrollo rápido de sitios web seguros y mantenibles. Desarrollado por programadores experimentados, Django se encarga de gran parte de las complicaciones del desarrollo web, por lo que puedes concentrarte en escribir tu aplicación sin necesidad de reinventar la rueda. Es gratuito y de código abierto, tiene una comunidad próspera y activa, una gran documentación y muchas opciones de soporte gratuito y de pago (MDM, 2019).

Características de Django:

Completo: Django sigue la filosofía "*baterías incluidas*" y provee casi todo lo que los desarrolladores quisieran que tenga "*de fábrica*". Porque todo lo que necesitas es parte de un único "*producto*", todo funciona a la perfección, sigue principios de diseño consistentes y tiene una amplia y actualizada documentación (MDM, 2019).

Versátil: Django puede ser usado para construir casi cualquier tipo de sitio web, desde sistemas manejadores de contenidos y wikis, hasta redes sociales y sitios de noticias. Puede funcionar con cualquier *framework* en el lado del cliente, y puede devolver contenido en casi cualquier formato. Internamente, mientras ofrece opciones para casi cualquier funcionalidad que desees (distintos motores de base de datos y motores de plantillas), también puede ser extendido para usar otros componentes si es necesario (MDM, 2019).

Seguro: Django ayuda a los desarrolladores evitar varios errores comunes de seguridad al proveer un *framework* que ha sido diseñado para "*hacer lo correcto*" para proteger el sitio web automáticamente. Por ejemplo, Django, proporciona una manera segura de administrar cuentas de usuario y contraseñas, evitando así errores comunes como colocar informaciones de sesión en cookies donde es vulnerable (en lugar de eso las cookies solo contienen una clave y los datos se almacenan en la base de datos) o se almacenan directamente las contraseñas en un hash de contraseñas (MDM, 2019).

Escalable: Django usa un componente basado en la arquitectura *"shared-nothing"* (cada parte de la arquitectura es independiente de las otras, y por lo tanto puede ser reemplazado o cambiado si es necesario). Teniendo en cuenta una clara separación entre las diferentes partes significa que puede escalar para aumentar el tráfico al agregar hardware en cualquier nivel: servidores de cache, servidores de bases de datos o servidores de aplicación. Algunos de los sitios más concurridos han escalado a Django para satisfacer sus demandas (MDM, 2019).

Mantenible: El código de Django está escrito usando principios y patrones de diseño para fomentar la creación de código mantenible y reutilizable. En particular, utiliza el principio No te repitas *"don't repeat yourself"* (DRY) para que no exista una duplicación innecesaria, reduciendo la cantidad de código. Django también promueve la agrupación de la funcionalidad relacionada en "aplicaciones" reutilizables y en un nivel más bajo, agrupa código relacionado en módulos (MDM, 2019).

Portable: Django está escrito en Python, el cual se ejecuta en muchas plataformas. Lo que significa que no está sujeto a ninguna plataforma en particular, y puede ejecutar sus aplicaciones en muchas distribuciones de Linux, Windows y Mac OS X. Además, Django cuenta con el respaldo de muchos proveedores de alojamiento web, y que a menudo proporcionan una infraestructura específica y documentación para el alojamiento de sitios de Django (MDM, 2019).

Para la presente propuesta de solución se escoge el marco de trabajo Django. La selección se realiza tomando en cuenta seguridad, escalabilidad, versatilidad, curva de aprendizaje entre otros.

Gestor de base de datos

PostgreSQL es un sistema de base de datos relacional de código abierto. Usa el lenguaje SQL combinado con muchas características que almacenan y escalan de manera segura las cargas de trabajo de datos más complicadas. Los orígenes de PostgreSQL se remontan a 1986 como parte del proyecto POSTGRES en la Universidad de California Berkeley y tiene más de 30 años de desarrollo activo en la plataforma central (Postgres, 2021).

PostgreSQL provee muchas características destinadas a ayudar a los desarrolladores a crear aplicaciones. También permite a los administradores proteger la integridad de los datos y a

crear entornos tolerantes a fallos. Además, proporciona administración de sus datos sin importar cuán grande o pequeño sea el conjunto. Es gratuito, de código abierto y altamente extensible. Intenta ajustarse al estándar SQL donde tal conformidad no contradice las características tradicionales. Se admiten muchas de las características requeridas por dicho estándar, aunque a veces con una sintaxis o función ligeramente diferente (PostgreSQL, 2021).

Lenguaje de programación

Python creado por Guido Van Rossum, es un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad (Reitz, 2018).

Entre sus características se encuentran:

Multiplataforma: Hay versiones disponibles de Python en muchos sistemas informáticos distintos. Originalmente se desarrolló para Unix, aunque cualquier sistema es compatible con el lenguaje siempre y cuando exista un intérprete programado para él (DesarrolloWeb.com, 2003).

Interpretado: Quiere decir que no se debe compilar el código antes de su ejecución. En realidad, sí que se realiza una compilación, pero esta se realiza de manera transparente para el programador (Reitz, 2018).

Orientado a Objetos: La programación orientada a objetos está soportada en Python y ofrece en muchos casos una manera sencilla de crear programas con componentes reutilizables(DesarrolloWeb.com, 2003).

Funciones y librerías: Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de *strings*, números, archivos, etc. Además, existen muchas librerías que podemos importar en los programas para tratar temas específicos como la programación de ventanas o sistemas en red o cosas tan interesantes como crear archivos comprimidos en .zip (Reitz, 2018).

Sintaxis clara: Python posee una sintaxis muy visual, gracias a una notación identada (con márgenes) de obligado cumplimiento. En muchos lenguajes, para separar porciones de código, se utilizan elementos como las llaves o las palabras clave ***begin*** y ***end***. Para separar las porciones de código en Python se debe tabular hacia dentro, colocando un margen al código que iría dentro de una función o un bucle. Esto ayuda a que todos los programadores adopten unas mismas notaciones y que los programas de cualquier persona tengan un aspecto muy similar (Reitz, 2018).

Entorno de desarrollo

JetBrains PyCharm es un IDE (IDE, por las siglas de *Integrated Development Environment*) es un entorno de programación que ha sido empaquetado como un programa de aplicación, o sea, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica, comercial y multiplataforma para Python, proporciona completamiento de código inteligente, inspecciones de código, resaltado de errores sobre la marcha y correcciones rápidas, junto con refactorizaciones de código automatizadas y ricas capacidades de navegación. Ofrece un gran soporte específico para marcos de desarrollo web modernos como *Django, Flask, Google App Engine, Pyramid y web2py*. Se integra con *IPython Notebook*, tiene una consola interactiva de Python y es compatible con Anaconda, así como con múltiples paquetes científicos como ***matplotlib*** y ***NumPy*** (JetBrains, 2020).

Lenguaje de modelado

El "*Lenguaje de Modelado Unificado*" (UML, por las siglas de *Unified Modeling Language*) es un lenguaje basado en diagramas cuyo objetivo general es su utilización para especificar, visualizar, construir y documentar los artefactos de un sistema de software. Captura decisiones y conocimiento sobre sistemas que deben ser construidos. Está pensado para ser utilizado con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre las técnicas de modelado e incorporar las mejores prácticas de *software* actuales, en una aproximación estándar. UML incluye conceptos semánticos, notación y principios generales (Rumbaugh et al., 2000).

Herramienta CASE

Para el uso de un lenguaje de modelado existen varias herramientas CASE (Ingeniería de Software Asistida por Computadoras), que son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y dinero. Estas herramientas, pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software, en tareas como el proceso de realizar un diseño del proyecto, cálculo del costo, compilación automática, documentación o detección de errores (Labañino Urbina et al., 2020). Para el desarrollo de la solución se propone utilizar el *Visual Paradigm*.

***Visual Paradigm*:** es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: Análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Ayuda a una rápida construcción de aplicaciones con calidad y a un menor costo. Permite crear todos los tipos de diagramas de clases, ingeniería inversa, generar código desde diagramas y generar documentación (Visual Paradigm, 2019).

1.6 Conclusiones parciales

- La informatización del control de personas que extinguen sanción en libertad es tarea clave para el proceso de aplicación de las TICs en el sistema judicial cubano.

- El estudio de soluciones similares como SISJE, eCourt, iJustice y Matrix arrojó como resultado que las soluciones actuales utilizadas y las soluciones foráneas no cubren las necesidades de la actividad del juez de ejecución.

- El estudio de varias metodologías como AUP, Crystal Metodologies, SCRUM y XP permitieron escoger XP para guiar el proceso de desarrollo de la propuesta.

- El estudio de distintos marcos de trabajo permitió obtener la selección de Django para el desarrollo de la solución, de la cual se obtiene como lenguaje de programación Python en su versión 3.10 y como gestor de base de datos PostgreSQL. Además, se selecciona PyCharm como IDE, como lenguaje de modelado UML y la herramienta CASE Visual Paradigm.

Capítulo 2. Diseño de la Propuesta de Solución

En el presente capítulo se presenta los usuarios con sus responsabilidades y la captura de requisitos con la elaboración de las historias de usuario junto a la planificación que plantea la metodología seleccionada. Además, se presentan los conceptos fundamentales en un diagrama conceptual del negocio, un diagrama de clases y el modelo entidad-relación. También se aborda sobre la arquitectura de software planteada por el ERP Odoo la cual se mantendrá para este proyecto.

2.1 Breve descripción de la propuesta de solución

La propuesta de solución tiene como objetivo apoyar a la actividad de control que llevan a cabo los jueces de ejecución en los tribunales municipales populares. Dentro del sistema los usuarios pueden desempeñarse como administrador o como asistente de ejecución (asistente).

Los usuarios que posean el rol de administrador serán los encargados del mantenimiento y correcto funcionamiento del sistema. Además, entre sus tareas se encuentran la gestión de usuarios, permisos, asistente de ejecución, provincias, municipios, consejos populares, tribunales, sanciones y delitos. Por otra parte, los usuarios con el rol de asistente serán los encargados de registrar a los sancionados en el sistema. Se encargará de la gestión de las visitas, asignar un juez de ejecución al sancionado, los traslados de los sancionados y los radicados sin documentos.

El sistema contará con una interfaz con los colores representativo del sistema de tribunales cubanos además del logo. Presentará interfaz intuitiva, fácil de recordar para los usuarios y sin mucha carga de información. La misma es una herramienta de apoyo para el trabajo que realiza el sistema de juez de ejecución.

2.2 Usuarios del sistema

Los usuarios del sistema son aquellas personas que interactúan con la aplicación. Para ellos existen algunas restricciones específicas. A continuación, se describen los usuarios del sistema con sus responsabilidades.

Tabla 3: Usuarios del sistema

Usuario	Responsabilidad
Administrador	Es el encargado de la gestión de los usuarios, así como la gestión de tribunales, provincia, municipios y delitos. Además del correcto funcionamiento del sistema.
Asistente	Es el encargado de la gestión de todo lo referente a los sancionados como adición, modificación y eliminación de los mismo, gestión de traslados, visitas y radicados sin documentos.

2.3 Requisitos Funcionales

Los requisitos son, por definición *"una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo"*, y que *"expresan las necesidades y limitaciones impuestas a un producto de software que contribuyen a la solución de algún problema en el mundo real".* La Ingeniería de Requisitos es tratada como una disciplina que tiene por propósito *"desarrollar una especificación de requerimientos completa, consistente y no ambigua, la cual servirá como base para acuerdos comunes entre todas las partes involucradas y en dónde se describen las funciones que realizará el sistema"* (Toro & Peláez, 2016).

RF1 Gestionar usuario

RF1.1 Añadir usuario

RF1.2 Listar usuario

RF1.3 Modificar usuario

RF1.4 Eliminar usuario

RF2 Gestionar sancionado

RF2.1 Añadir sancionado

RF2.2 Listar sancionado

RF2.3 Modificar sancionado

RF2.4 Eliminar sancionado

RF3 Gestionar delito

RF3.1 Añadir delito

RF3.2 Listar delito

RF3.3 Modificar delito

RF3.4 Eliminar delito

RF4 Gestionar sanción

RF4.1 Añadir sanción

RF4.2 Listar sanción

RF4.3 Modificar sanción

RF4.4 Eliminar sanción

RF5 Gestionar tribunal

RF5.1 Añadir tribunal provincial popular

RF5.2 Listar tribunal provincial popular

RF5.3 Modificar tribunal provincial popular

RF5.4 Eliminar Tribunal Provincial Popular

RF6 Gestionar provincia

RF6.1 Añadir provincia

RF6.2 Listar provincia

RF6.3 Modificar provincia

RF6.4 Eliminar provincia

RF7 Gestionar municipio

RF7.1 Añadir municipio

RF7.2 Listar municipio

RF7.3 Modificar municipio

RF7.4 Eliminar municipio

RF8 Gestionar traslado

RF8.1 Añadir traslado

RF8.2 Listar traslado

RF8.3 Modificar traslado

RF8.4 Eliminar traslado

RF9 Gestionar consejo popular

RF9.1 Añadir consejo popular

RF9.2 Listar consejo popular

RF9.3 Modificar concejo popular

RF9.4 Eliminar consejo popular

RF10 Gestionar radicado sin documento

RF10.1 Añadir radicado sin documento

RF10.2 Listar radicado sin documento

RF10.3 Eliminar radicado sin documento

RF11 Gestionar visita

RF11.1 Añadir visita

RF11.2 Listar visita

RF11.3 Modificar visita

RF11.4 Eliminar visita

RF12 Gestionar asistente de ejecución

RF12.1 Añadir asistente de ejecución

RF12.2 Listar asistentes de ejecución

RF12.3 Modificar asistente de ejecución

RF12.4 Eliminar asistente de ejecución

RF13 Generar reporte estadístico

2.4 Requisitos No Funcionales

Usabilidad

- **RNF1** Interfaz amigable fáciles de usar por el usuario, donde el destino final dentro del sistema no se encuentre a más de tres clics.
- **RNF2** El sistema debe proporcionar mensajes de error que sean informativos y orientados a usuario final.
- **RNF3** El sistema debe poseer interfaces gráficas bien formadas, los elementos de formularios descritos y navegación intuitiva.

Seguridad

- **RNF4** El acceso debe estar restringido por usuario y contraseña, las contraseñas deben estar cifradas en la base de datos.
- **RNF5** El sistema debe asegurar que los datos estén protegidos del acceso no autorizado según el rol.

Software

- **RNF6** En el cliente el navegador web debe ser Mozilla Firefox 65 o superior.

Apariencia

- **RNF7** El sistema debe contener los colores institucionales y con presencia visible del logo de la institución.

- **RNF8** El sistema debe ser interactivo e indicar el estado actual de la tarea que realiza el usuario sobre este.

Accesibilidad

- **RNF9** El sistema debe estar disponible desde cualquier estación de trabajo conectada a la red y permitir su administración de forma remota.

2.5 Historias de usuarios

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. En ellas el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden ser eliminadas, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas (Bruzón, 2019).

Tabla 4: HU2- Gestionar sancionado

Historia de usuario	
Número: 2	**Nombre:** Gestionar sancionado
Usuarios: Asistente	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Alto
Estimación: 5 días	**Iteración Asignada:** 2
Descripción: El asistente gestiona todo lo referente a los sancionados del sistema.	
Observaciones: El usuario se encargará de adicionar, listar, modificar y eliminar sancionados.	

Tabla 5: HU3- Gestionar delito

Historia de usuario	
Número: 3	**Nombre:** Gestionar delito
Usuarios: Administrador	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Estimación: 2 días	**Iteración Asignada:** 1
Descripción: El administrador gestiona todo lo referente a los delitos establecidos en el código penal de la República de Cuba.	
Observaciones: El usuario se encargará de adicionar, listar, modificar y eliminar delitos.	

Tabla 6: HU4- Gestionar sanción.

Historia de usuario	
Número: 4	**Nombre:** Gestionar sanción
Usuarios: Administrador	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Estimación: 2 días	**Iteración Asignada:** 1
Descripción: El usuario se encargará de todo lo referente a la gestión de las sanciones previstas en el código penal de la República de Cuba.	
Observaciones: El usuario se encargará de adicionar, listar, modificar y eliminar sanciones.	

Tabla 7: HU8- Gestionar traslado.

Historia de usuario	
Número: 8	**Nombre:** Gestionar traslado
Usuarios: Asistente	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Alta
Estimación: 4 días	**Iteración Asignada:** 2
Descripción: El asistente es el encargado de realizar todo lo referente con el traslado de un sancionado.	
Observaciones: El usuario podrá añadir, listar y modificar los traslados de los sancionados. En caso que se de baja al sancionado del sistema automáticamente se eliminan los traslados.	

2.6 Planificación

Todo proyecto que se guía por la metodología XP, se planifica en iteraciones de aproximadamente tres semanas de duración. Para cada iteración se define un conjunto de HU que se van a implementar (Acosta & Martínez, 2015).

Tabla 8: Plan de Iteraciones.

Iteraciones	Orden de las HU a Implementar	Tiempo de trabado
1	Gestionar Usuario, Gestionar Delito, Gestionar Sanción, Gestionar Tribunal, Gestionar Provincia, Gestionar Municipio, Gestionar Consejo Popular, Gestionar Asistente de Ejecución	4 semanas
2	Gestionar Sancionado, Gestionar Traslado, Gestionar Radicado sin Documento, Gestionar Visitas	3 semanas
3	Gestionar Reportes Estadísticos	3 semanas

El plan de entregas es el compromiso final del equipo de trabajo con los clientes. Esta es una cuestión de vital importancia para ambas partes, ya que la entrega tardía o temprana de la solución, repercute notablemente en la economía y la moral de los implicados (Abreu, 2019).

Tabla 9: Plan de Entrega.

Plan de Entrega		
Iteración 1	Iteración 2	Iteración 3
16 de octubre de 2021	10 de noviembre de 2021	31 de noviembre de 2021

2.7 Diseño

Un diagrama conceptual del negocio no es más que un artefacto construido bajo las reglas de UML durante la concepción de un proyecto informático. Este modelo puede ser utilizado para capturar y expresar el entendimiento ganado sobre el negocio (Acosta & Martínez, 2015).

Se determinó necesario generar este artefacto para mejor entendimiento del negocio. Así identificando los principales conceptos y como se relacionan entre ellos facilita el desarrollo del de la aplicación web.

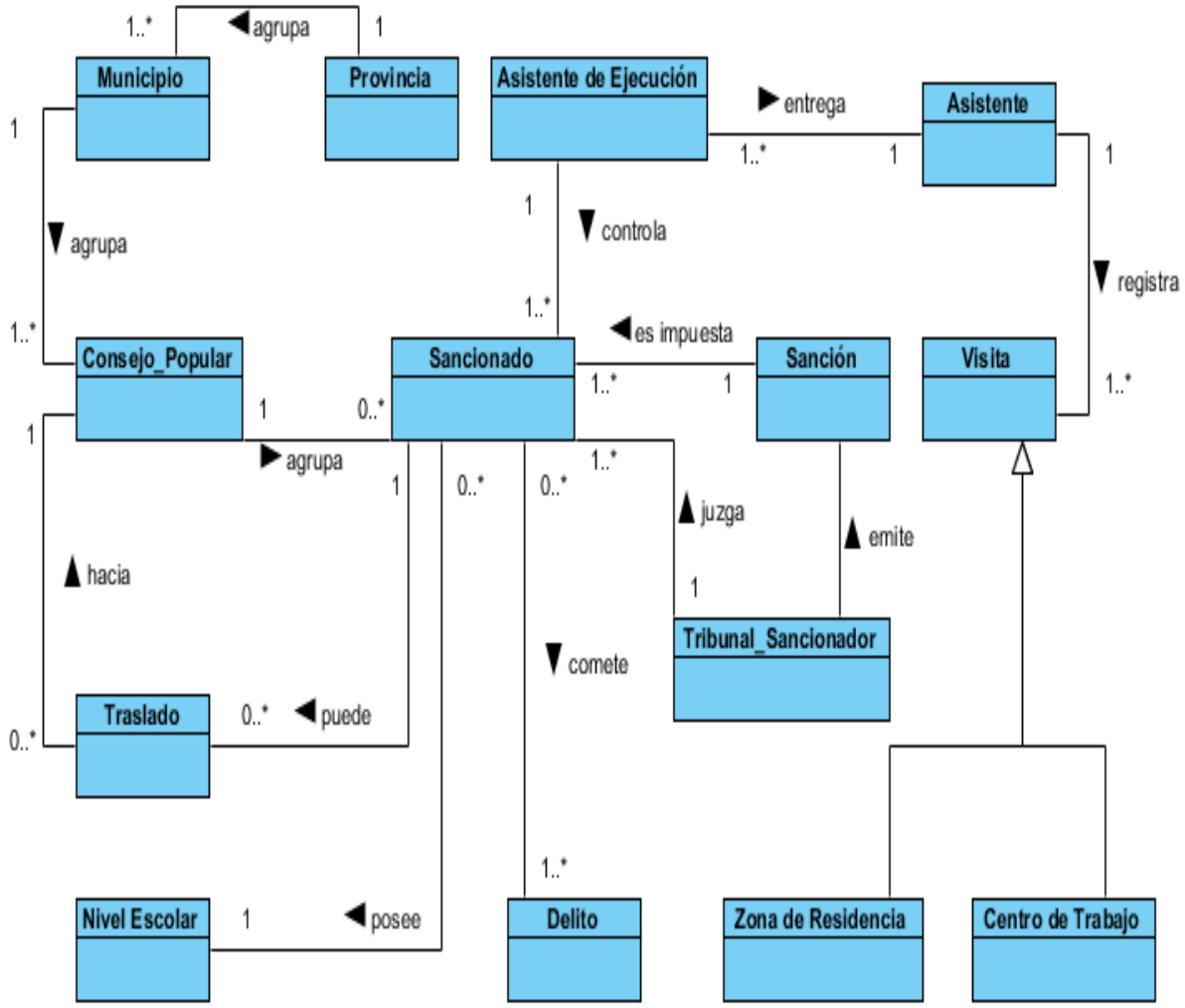

Figura 3: Modelo Conceptual del Negocio

Diagrama de clase

El diagrama de clases describe gráficamente las especificaciones de las clases de software en una aplicación. A diferencia del modelo conceptual, un diagrama de este tipo contiene las definiciones de las entidades del software en vez de conceptos del mundo real. El UML no define concretamente un elemento denominado "*diagrama clases del diseño*", sino que se sirve de un término más genérico: "*diagrama de clases*". Se trata de una perspectiva desde el punto de vista del diseño de las entidades de software y no de una concepción analítica sobre los conceptos del dominio (Larman, 2003).

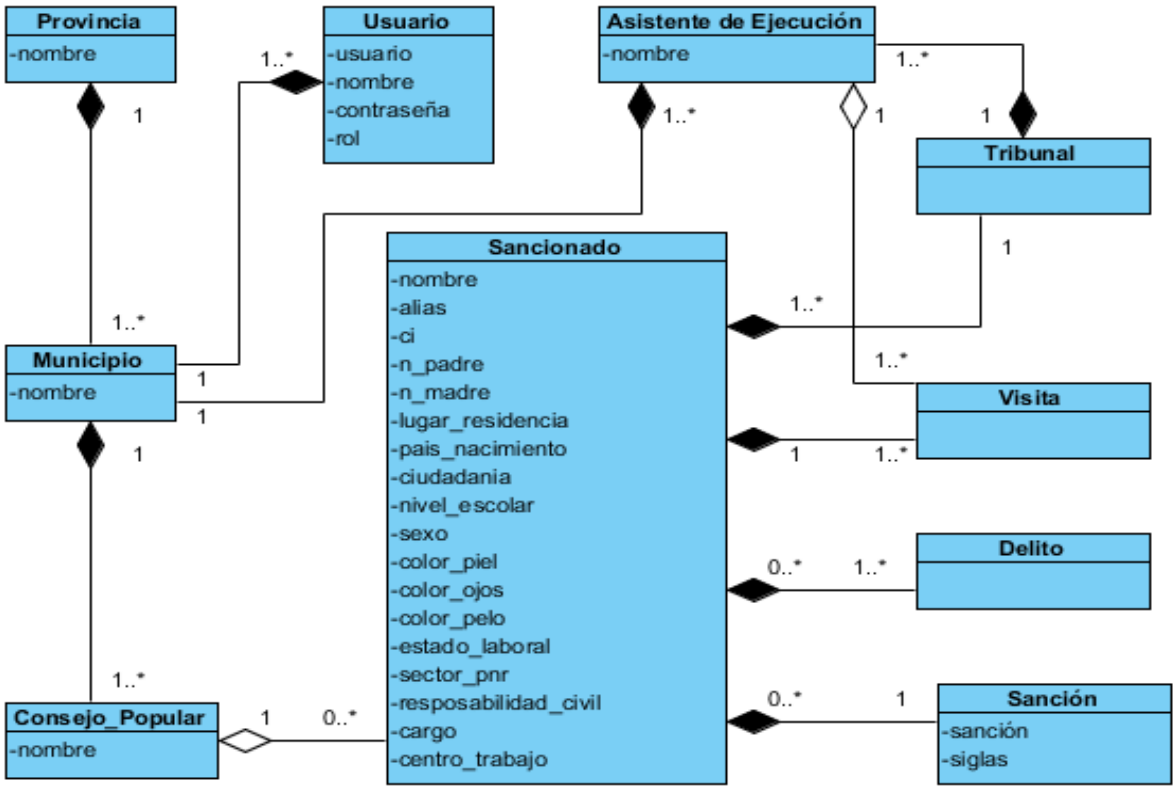

Figura 4:Diagrama de Clases

Modelo Entidad-Relación

El Modelo Entidad Relación (ER) permite desarrollar un diseño de base de datos en un esquema de alto nivel conceptual sin considerar los problemas de bajo nivel como la eficiencia, el modelo implícito del administrador de base de datos o las estructuras físicas de los datos. El Modelo Entidad Relación se hizo muy popular para el diseño de base de datos y es usado extensivamente. Para aumentar su poder de expresión, muchos investigadores han introducido o propuesto ciertas extensiones a este modelo. Algunas de estas extensiones son

importantes, mientras que otras agregan poco poder de expresión, pero proveen características auxiliares. Puesto que el Modelo Entidad Relación es ampliamente usado, es importante conocer qué extensiones han sido propuestas para este modelo y qué ofrecen estas extensiones a los usuarios (Saiedian, 1997).

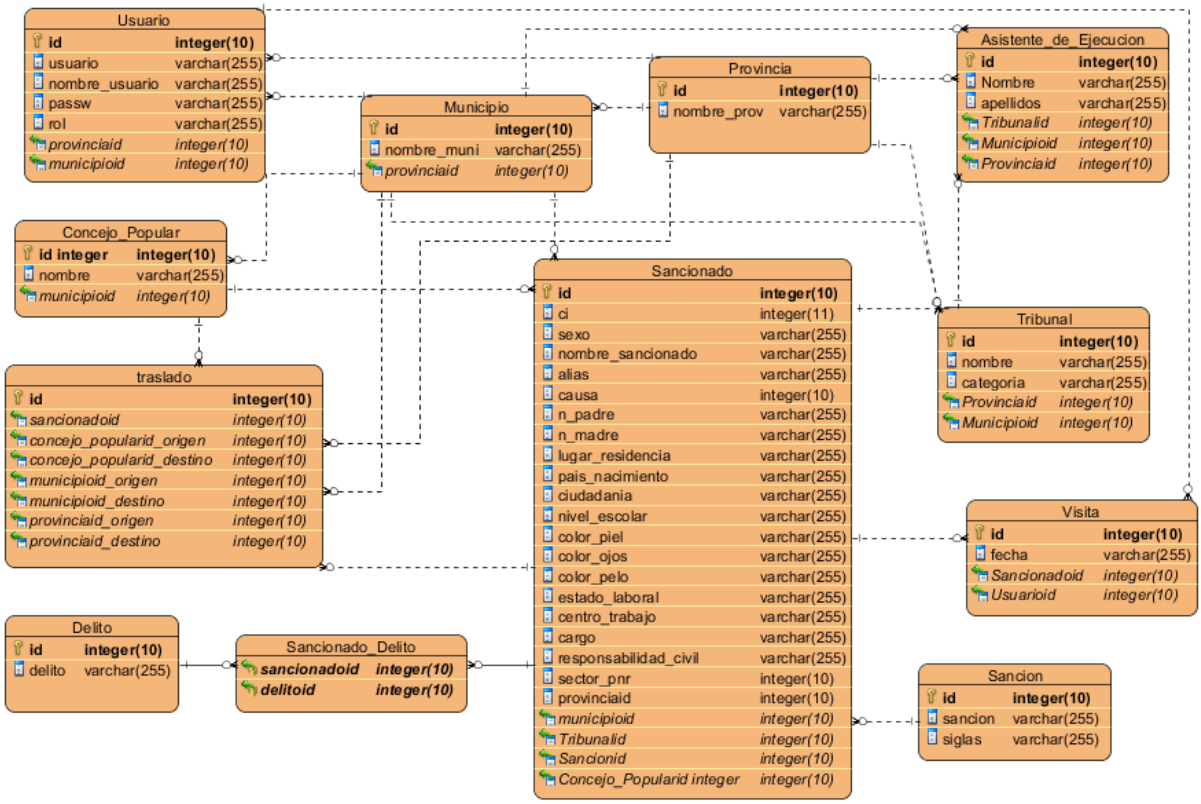

Figura 5: Modelo Entidad-Relación

2.8 Descripción de la arquitectura

Los sistemas de software dependen directamente de su diseño arquitectónico para garantizar su uso a largo plazo, un mantenimiento eficiente y una evolución adecuada en un entorno de ejecución. Diversos autores afirman que las arquitecturas son el factor crítico en la capacidad de los sistemas de software para perdurar y evolucionar (Venters et al., 2017). La creatividad de los arquitectos de software puede llevar a diferentes diseños o arquitecturas de soluciones que sirvan para el mismo propósito. Los diagramas que ilustran los diseños propuestos pueden valer miles de palabras (Jansen et al., 2020).

Arquitectura Modelo Vista Plantilla (MVT)

Diseñado para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Sus características principales están dadas por el hecho de que, el modelo, las vistas y las plantillas se tratan como entidades separadas. Esto hace que cualquier cambio producido en el modelo se refleje de forma automática en cada una de las vistas (Romero & González, 2012).

Sin embargo, en el momento en que aparecen las bases de datos y los modelos, este proceso se extiende. Ahora se recibirá la petición, se pasará a la vista, en la vista recuperaremos los datos del modelo correspondiente, y finalmente la renderizaremos el Plantilla, pero esta vez integrando los datos dinámicos recuperados del modelo, antes de enviar el HTML final al navegador (Costa, 2019):

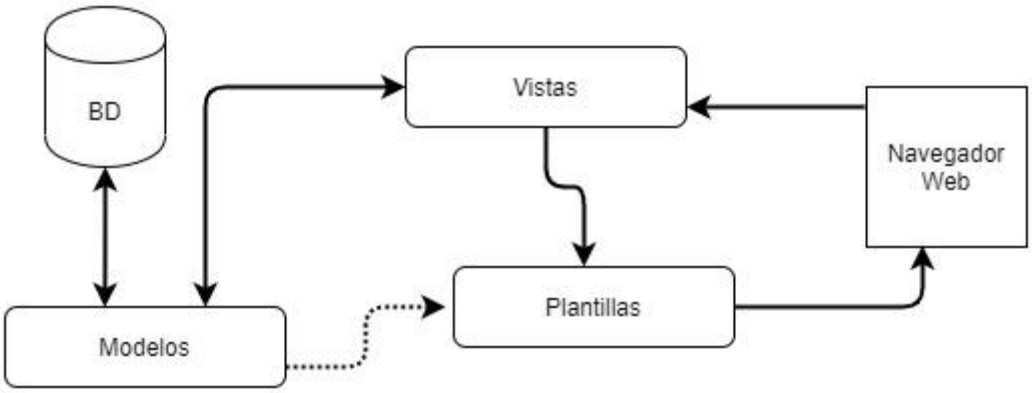

Figura 6: Modelo-Vista-Template

Modelo: Contiene una representación de los datos que maneja el sistema. Define las reglas de negocio (que es lo que va a hacer el sistema). Es el encargado de acceder a la capa de almacenamiento de datos (Caccia, 2019).

Vista: Son responsables de recibir datos procesados por el controlador o del modelo y mostrarlos al usuario. Tienen un registro de su controlador asociado. Pueden dar el servicio de "Actualización ()", para que sea invocado por el controlador o por el modelo cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes (Caccia, 2019).

Plantilla: Actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de

cada uno. Recibe los eventos de la entrada (un clic, un cambio en un campo de texto, etc). Define la lógica de presentación, mediante manejadores de eventos (Caccia, 2019).

2.9 Patrones de diseño de software

Los patrones de diseño facilitan la reutilización de diseños y arquitecturas exitosas. Un patrón de diseño sistemáticamente nombra, motiva y explica un diseño general que aborda un problema recurrente de diseño en los sistemas orientados a objetos. Describe el problema, la solución, cuándo aplicar la solución y sus consecuencias. También brinda consejos de aplicación y ejemplos. La solución es una disposición general de objetos y clases que resuelven el problema. La misma se adapta y se aplica para resolver el problema en un contexto particular (Labañino Urbina et al., 2020).

Patrones GRASP

Los patrones GRASP (por sus siglas en inglés, General Responsibility Assignment Software Patterns) describen los principios fundamentales de la asignación de responsabilidades a objetos. El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos (Demestre & González, 2019).

Los patrones GRASP utilizados en la solución propuesta fueron:

- **Creador:** se emplea cuando se le aplica la responsabilidad a una clase determinada de crear una o más instancias de otra. Esto sucede en caso de que la clase creadora contenga, agregue, registre, utilice o posea datos de inicialización de objetos de alguna clase determinada (Larman, 2003). La utilización de este patrón es muy importante para las clases encargadas de registrar datos de usuarios en el sistema, como lo hace la clase ***Provincia, Sancionado*** entre otras que registran información en la base de datos.

- **Controlador:** asigna la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión (Larman, 2003). Este patrón se evidencia en la en ***SancionadoCreateView.***

- **Bajo acoplamiento:** debe haber pocas dependencias entre las clases. El propósito de este patrón es aumentar la reutilización y eliminar las dependencias entre las clases para propiciar un fácil mantenimiento y entendimiento (Larman, 2003).

- **Alta cohesión:** cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. Este patrón permitirá al equipo de desarrollo que la implementación de las clases encargadas de codificar la información sea fácil de comprender, reutilizar, mantener y poco susceptibles a cambios. En el sistema implementado se agruparon las clases según la funcionalidad de cada una (Larman, 2003).

Patrones GOF

Publicados por Gamma, Helm, Johnson y Vlossodes en 1995: patrones de la banda de los cuatro (del inglés, Gang of Four). Esta serie de patrones permiten ampliar el lenguaje, aprender nuevos estilos de diseño y además introducir más notación UML. Los patrones de diseño del grupo GoF se clasifican en tres grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento (Demestre & González, 2019). A continuación, se describe el patrón GoF utilizado en la solución propuesta:

Decorador: Patrón estructural que extiende la funcionalidad de un objeto dinámicamente de manera tal que es transparente a sus clientes, utiliza una instancia de una subclase de la clase original que delega las operaciones al objeto original (Demestre & González, 2019). Este patrón se evidencia en la clase ***Login_require*** (decorador que trae Django por defecto, para acceder a una clase)

2.10 Conclusiones parciales

- La realización del Modelo conceptual permitió una mejor comprensión del negocio, debido a que se manejan los conceptos fundamentales que involucran los procesos de la actividad de atención y control a personas que extinguen sanción en libertad.

- De la propuesta de solución planteada se detectaron un total de 49 funcionalidades del sistema y 9 características no funcionales con las que debe contar el mismo para satisfacer las necesidades del cliente.

- Luego de culminar la fase de planificación definida por la metodología seleccionada, quedó pactado con el cliente que se realizarán tres iteraciones y el tiempo máximo entre las mismas será de cuatro semanas.

- La utilización de los patrones de diseño GRASP y GOF permitieron tener una guía para la fase posterior, contribuyendo al uso de buenas prácticas en el proceso de modelado e implementación del software.

Capítulo 3. Implementación y Pruebas

En el presente capítulo se aborda acerca de la arquitectura de software utilizada en la propuesta de solución. También se muestran las tareas de ingeniería utilizadas para la implementación de la aplicación web. Además de se expone el proceso de prueba al que es sometido el software.

3.1 Tareas de ingeniería

XP propone dividir cada HU en tareas de ingeniería con el objetivo de facilitar la implementación a los programadores, estas tareas deben tener una duración de entre uno y tres días aproximadamente. Las 12 HU redactadas fueron desglosadas en un total de 50 tareas de ingeniería, en este epígrafe solo se muestran las relacionadas con las HU que se describen en el Capítulo 2, las restantes se encuentran en el Anexo 2 de la investigación.

3.1.1 Iteración 1

En esta iteración, el objetivo es crear la aplicación con la base y estructura necesaria para que los usuarios del sistema puedan empezar a insertar sancionados al sistema. Las tareas a realizar son las siguientes:

Tarea 1: Adicionar usuario

Tarea 2: Modificar usuario

Tarea 3: Listar usuarios

Tarea 4: Eliminar usuario

Tarea 5: Adicionar delito

Tarea 6: Modificar delito

Tarea 7: Listar delitos

Tarea 8: Eliminar delito

Tarea 9: Adicionar sanción

Tarea 10: Modificar sanción

Tarea 11: Listar sanción

Tarea 12: Eliminar sanción

Tarea 13: Adicionar tribunal

Tarea 14: Modificar tribunal

Tarea 15: Listar tribunal

Tarea 16: Eliminar tribunal

Tarea 17: Adicionar provincia

Tarea 18: Modificar provincia

Tarea 19: Listar provincia

Tarea 20: Eliminar provincia

Tarea 21: Adicionar municipio

Tarea 22: Modificar municipio

Tarea 23: Listar municipio

Tarea 24: Eliminar municipio

Tarea 25: Adicionar concejo popular

Tarea 26: Modificar concejo popular

Tarea 27: Listar concejo popular

Tarea 28: Eliminar concejo popular

Tarea 29: Adicionar juez de ejecución

Tarea 30: Modificar juez de ejecución

Tarea 31: Listar juez de ejecución

Tarea 32: Eliminar juez de ejecución

Tabla 10: Tarea de ingeniería 1. Adicionar usuario

Tarea	
Número de Tarea: 1	**Número de la Historia de usuario: 1**
Nombre de la Tarea: Adicionar Usuario	

Tipo de Tarea: Desarrollo	**Estimación:** 1 día
Fecha de inicio: 16-10-21	**Fecha de fin:** 16-10-21
Programador responsable: Hugo Alberto Valencia Zayas	
Descripción: Implementar una vista que permita adicionar un usuario al sistema.	

Tabla 11: Tarea de ingeniería 2. Modificar usuario

Tarea	
Número de Tarea: 2	**Número de la Historia de usuario:** 1
Nombre de la Tarea: Modificar Usuario	
Tipo de Tarea: Desarrollo	**Estimación:** 1 día
Fecha de inicio: 17-10-21	**Fecha de fin:** 17-10-21
Programador responsable: Hugo Alberto Valencia Zayas	
Descripción: Implementar una vista que permita editar un usuario del sistema.	

Tabla 12: Tarea de ingeniería 3. Listar usuario

Tarea	
Número de Tarea: 3	**Número de la Historia de usuario:** 1
Nombre de la Tarea: Listar Usuario	
Tipo de Tarea: Desarrollo	**Estimación**: 1 día
Fecha de inicio: 18-10-21	**Fecha de fin:** 18-10-21
Programador responsable: Hugo Alberto Valencia Zayas	
Descripción: Implementar una vista que permita listar los usuarios del sistema.	

Tabla 13: Tarea de ingeniería 4. Eliminar usuario

Tarea	
Número de Tarea: 4	**Número de la Historia de usuario:** 1
Nombre de la Tarea: Eliminar Usuario	
Tipo de Tarea: Desarrollo	**Estimación: 1 día**
Fecha de inicio: 19-10-21	**Fecha de fin:** 19-10-21
Programador responsable: Hugo Alberto Valencia Zayas	
Descripción: Implementar una vista que permita eliminar un usuario al sistema.	

3.1.2 Iteración 2

En esta iteración, el objetivo es adicionar las funcionalidades de peso de la propuesta de solución. Las tareas a realizar son las siguientes:

Tarea 33: Adicionar sancionado

Tarea 34: Modificar sancionado

Tarea 35: Listar sancionado

Tarea 36: Eliminar sancionado

Tarea 37: Adicionar traslado

Tarea 38: Modificar traslado

Tarea 39: Listar traslado

Tarea 40: Eliminar traslado

Tarea 41: Adicionar visita

Tarea 42: Listar visita

Tarea 43: Eliminar visita

Tarea 44: Adicionar radicado sin documento

Tarea 45: Modificar radicado sin documento

Tarea 46: Listar radicados sin documento

Tarea 47: Eliminar radicados sin documento

Tabla 14: Tarea de ingeniería 33. Adicionar sancionado

Tarea	
Número de Tarea: 33	**Número de la Historia de usuario: 2**
Nombre de la Tarea: Adicionar Sancionado	
Tipo de Tarea: Desarrollo	**Estimación:** 1 día
Fecha de inicio: 10-11-21	**Fecha de fin:** 10-11-21

Programador responsable: Hugo Alberto Valencia Zayas
Descripción: Implementar una vista que permita adicionar un sancionado al sistema.

Tabla 15: Tarea de ingeniería 34. Modificar sancionado

Tarea	
Número de Tarea: 34	**Número de la Historia de usuario: 2**
Nombre de la Tarea: Modificar Sancionado	
Tipo de Tarea: Desarrollo	**Estimación: 1 día**
Fecha de inicio: 11-11-21	**Fecha de fin:** 11-11-21
Programador responsable: Hugo Alberto Valencia Zayas	
Descripción: Implementar una vista que permita modificar un sancionado al sistema.	

Tabla 16: Tarea de ingeniería 35. Listar sancionados

Tarea	
Número de Tarea: 35	**Número de la Historia de usuario: 2**
Nombre de la Tarea: Listar Sancionados	
Tipo de Tarea: Desarrollo	**Estimación:** 1 día
Fecha de inicio: 12-11-21	**Fecha de fin:** 12-11-21
Programador responsable: Hugo Alberto Valencia Zayas	
Descripción: Implementar una vista que permita listar los sancionados del sistema.	

Tabla 17: Tarea de Ingeniería 36. Eliminar Sancionado

Tarea	
Número de Tarea: 36	**Número de la Historia de usuario: 2**
Nombre de la Tarea: Eliminar Sancionados	
Tipo de Tarea: Desarrollo	**Estimación:** 1 día
Fecha de inicio: 13-11-21	**Fecha de fin:** 13-11-21
Programador responsable: Hugo Alberto Valencia Zayas	
Descripción: Implementar una vista que permita eliminar un sancionado al sistema.	

3.1.3 Iteración 3

En esta iteración, el objetivo es crear un soporte de información estadístico mediante graficas que ayuden a comprender la información contenida. Las tareas a realizar son las siguientes:

Tarea 48: Graficar datos por sexo.

Tarea 49: Graficar datos por grupos etarios.

Tarea 50: Graficar datos por delitos.

Tarea 51: Graficar datos por concejos populares.

3.2 Pruebas

La metodología XP enfatiza en la realización de pruebas a lo largo de todo el desarrollo del software, con el fin de lograr un producto con calidad, reduciendo el número de errores y disminuyendo el tiempo transcurrido entre la aparición de un error y su corrección. En este proceso no solo participa el equipo de desarrollo, también es importante la colaboración del cliente, sobre todo en las pruebas de aceptación (Acosta & Martínez, 2015).

3.2.1 Desarrollo dirigido por pruebas (TDD)

Desarrollo Orientado a Pruebas es el desarrollo de software orientado a objetos que incluye un ciclo de pruebas constantes al software, se emplea en las metodologías ágiles, en las cuales se escribe código para superar las pruebas que se han especificado con anterioridad, y en algunas ocasiones sustituyen la especificación de requisitos (Jaramillo, 2015).

Los pasos dentro del enfoque TDD son los siguientes (Roman & Mnich, 2021):

1. Escribir una prueba para la funcionalidad que se va a implementar.

2. Ejecutar la prueba (la nueva prueba debe fallar, porque no hay código para ella) - este paso verifica que las pruebas están escritas correctamente.

3. Implementar la cantidad mínima de código para que todas las pruebas pasen - este paso verifica que el código implementa la funcionalidad prevista para una iteración determinada. En caso de fallo modificar el código hasta que se superen todas las pruebas.

4. Refactorizar el código para mejorar su legibilidad y mantenimiento.

5. Vuelva al paso 1.

3.2.2 Pruebas unitarias

En la presente investigación se utilizó el nivel de pruebas unitarias utilizando la técnica de caja blanca. Estas pruebas permiten determinar si un módulo del sistema está listo y correctamente terminado. A la hora de aplicarlas se tienen en cuenta algunos parámetros, los cuales se muestran a continuación:

- Los datos de entrada son conocidos por el Encargado de Pruebas y estos deben ser preparados con minuciosidad, puesto que el resultado de las pruebas depende de estos.
- Se debe conocer qué componentes interactúan en cada caso de prueba.
- Se debe conocer de antemano, qué resultados debe devolver el componente según los datos de entrada utilizados en la prueba.

Finalmente se deben comparar los datos obtenidos en la prueba con los datos esperados, si son idénticos el módulo supera la prueba (Acosta & Martínez, 2015).

Usando la estructura del framework Django, el cual crea un archivo test.py y haciendo uso de la biblioteca de Python TestCase se realizaron las pruebas unitarias. A continuación, se muestra las primeras líneas necesarias para la prueba (Figura 7).

```
1  from django.test import TestCase
2  from juez_ejecucion.models import Provincia
3  from django.urls import reverse
4
```

Figura 7: Bibliotecas para pruebas unitarias.

Ya teniendo importado la biblioteca y el modelo a probar se procede a crear la clase de prueba. Dentro contiene las funciones encargadas de hacer los test (Figura 8).

```python
class ProvinciaModelTest(TestCase):

    @classmethod
    def setUpTestData(cls):
        Provincia.objects.create(nombre='Habana')

    def setUp(self):
        print("[<Prueba> : Modelo Provincia]")
        pass

    def test_name(self):
        provincia=Provincia.objects.get(id=1)
        nombre = provincia.nombre
        self.assertEquals(nombre,'Habana')
        print("[<Prueba> : Modelo Provincia <Succesfully>]")

```

Figura 8: Prueba unitaria al modelo Provincia

También se realizan test a los formularios y a las vistas. A continuación, se muestra una prueba realizada a la vista ***provincia_view.py*** (Figura 9).

```python
class ProvinciaListViewTest(TestCase):

    @classmethod
    def setUpTestData(cls):
        print("[<Prueba> : Vista provincia_view]")
        #Creando 10 provincias para prueba
        number_of_province = 10
        for province_num in range(number_of_province):
            Provincia.objects.create(nombre='Provincia %s' % province_num)

    def test_view_url_accessible_by_name(self):
        resp = self.client.get(reverse('juez_ejecucion:provincialist'))
        self.assertEqual(resp.status_code, 200)
        print("[<Prueba> : Vista povincia_view <Succesfully>]")

```

Figura 9: Prueba unitaria a la vista provincia_view.py.

Para ejecutar las pruebas se realiza mediante la consola con el comando manage.py test donde se ejecutan todas las pruebas del archivo ***test.py***. En la consola se puede observar cuantos casos de pruebas son realizados, cuantos fallan y en caso de no ocurrir fallos se muestra un OK. A continuación, se muestra un caso donde ocurre un fallo (Figura 10).

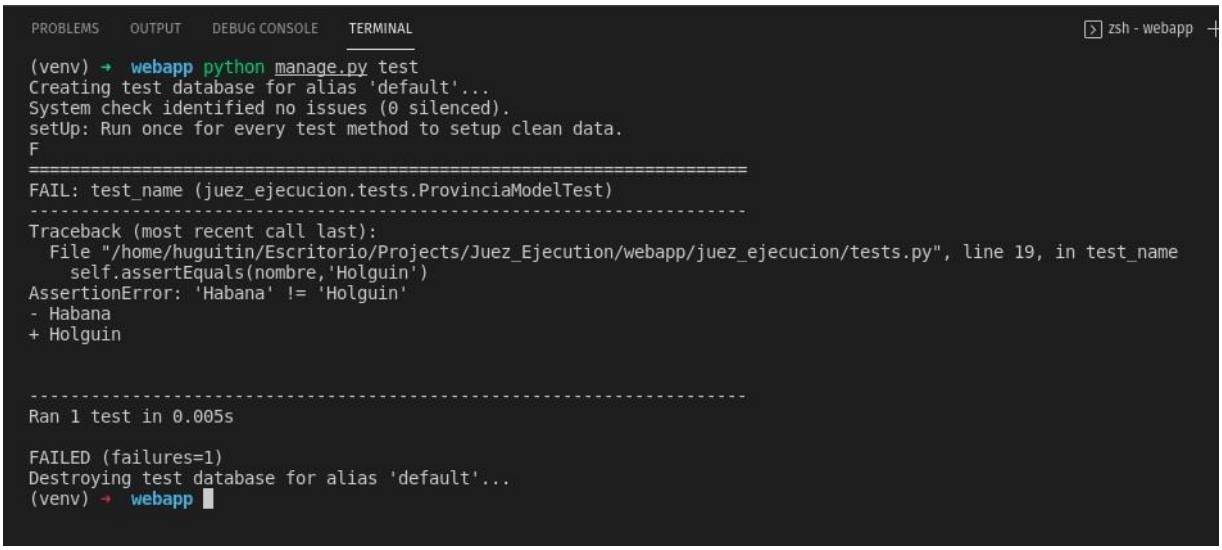

Figura 10: Resultado prueba unitaria 1

En caso de todo funcionar correctamente al terminar todas las pruebas realizadas se observa como en la siguiente imagen (Figura 11).

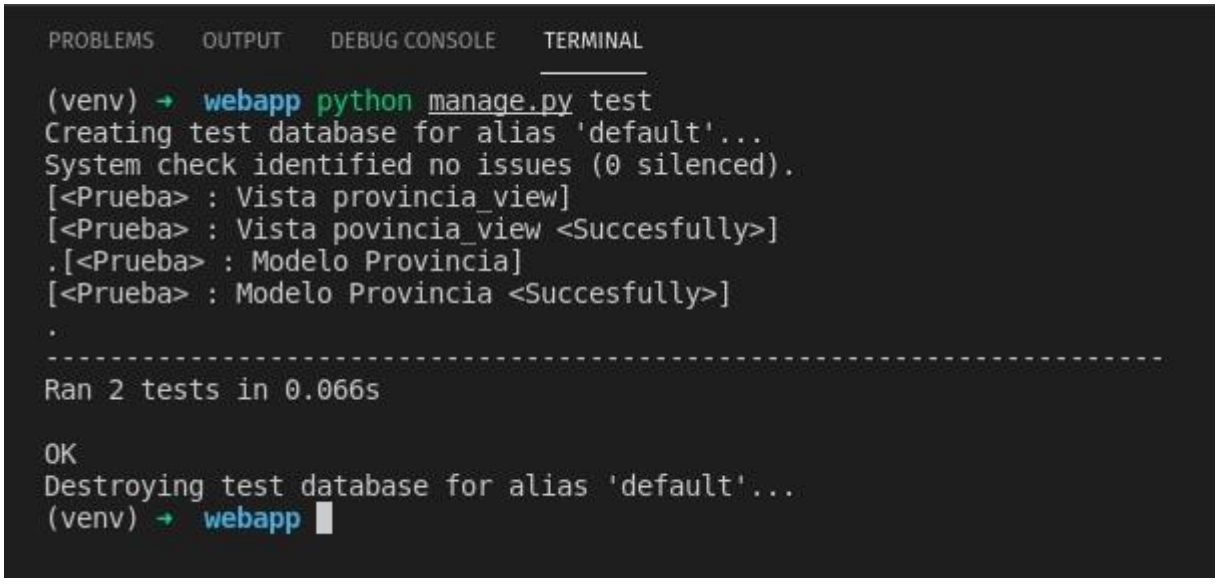

Figura 11: Resultado prueba unitaria 2

3.2.3 Pruebas de aceptación

Para realizarle pruebas a las funcionalidades del sistema se utilizó el nivel de pruebas de aceptación utilizando la técnica de caja negra. Las pruebas de aceptación son las especificaciones para el comportamiento deseado y la funcionalidad de un sistema. Estas pruebas fueron escritas por el equipo de desarrollo en conjunto con el cliente antes de

desarrollar cada HU y aplicadas al final de cada iteración. A continuación, se presentan los casos de pruebas de aceptación pertenecientes a las HU mostradas en el Capítulo 2.

Tabla 18: Prueba de aceptación 1

Caso de prueba de aceptación	
Código: HU2_P1	**Historia de usuario: 2**
Nombre: Listar sancionados	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema con el rol de asistente.	
Entrada/Pasos de ejecución: Hacer clic en Sancionados donde son listados los existentes.	
Resultados esperados: Una vista con todos los usuarios listados.	
Evaluación de la prueba: Satisfactoria	

Tabla 19: Prueba de aceptación 2

Caso de prueba de aceptación	
Código: HU2_P2	**Historia de usuario:** 2
Nombre: Añadir sancionado	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema con el rol de asistente para añadir un sancionado.	
Entrada/Pasos de ejecución: Hacer clic en Sancionados donde son listados los existentes, presionar “Añadir”, rellenar los datos y presionar “Guardar”.	
Resultados esperados: El usuario es añadido al sistema.	
Evaluación de la prueba: Satisfactorio	

Tabla 20: Prueba de aceptación 3

Caso de prueba de aceptación	
Código: HU3_P3	**Historia de usuario:** 3

Nombre: Añadir delito
Condiciones de ejecución: El usuario debe estar autenticado en el sistema con el rol de administrador para añadir un delito.
Entrada/Pasos de ejecución: Hacer clic en delitos donde son listados los existentes, presionar “Añadir”, rellenar los datos y presionar “Guardar”.
Resultados esperados: El delito es añadido al sistema.
Evaluación de la prueba: Satisfactorio

Tabla 21: Prueba de aceptación 4

Caso de prueba de aceptación	
Código: HU3_P4	**Historia de usuario:** 3
Nombre: Modificar delito	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema con el rol de administrador para modificar un delito.	
Entrada/Pasos de ejecución: Hacer clic en Delitos donde son listados los existentes, presionar el icono de editar, modificar los datos deseados y presionar “Guardar”.	
Resultados esperados: El delito es modificado.	
Evaluación de la prueba: Satisfactorio	

Tabla 22: Prueba de aceptación 5.

Caso de prueba de aceptación	
Código: HU3_P5	**Historia de usuario:** 3
Nombre: Eliminar delito	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema con el rol de administrador para añadir un delito.	

Entrada/Pasos de ejecución: Hacer clic en Delitos donde son listados los existentes, presionar el icono de eliminar.
Resultados esperados: El delito es eliminado del sistema.
Evaluación de la prueba: Satisfactorio

3.2.4 Análisis de los resultados de las pruebas de aceptación

Al concluir las tres iteraciones planificadas para el desarrollo de la propuesta de solución, fueron realizadas las pruebas pertinentes para las entregas pactadas con el cliente, la Tabla 22 muestra el resultado de estas pruebas en cada una de las iteraciones. Todas las No Conformidades fueron resueltas, lo cual, valida la calidad de la propuesta de solución logrando una mayor satisfacción por parte del cliente.

Tabla 23: Cantidad de NC por iteraciones

Iteraciones	**No Conformidades detectadas**			
	Alta	**Media**	**Baja**	**Total**
1	6	8	3	17
2	3	5	2	10
3	0	1	3	4

3.3 Conclusiones parciales

- La división de las historias de usuario en tareas de ingeniería le indicó al equipo de desarrollo los objetos que se deben construir obteniendo como resultado un excelente flujo de trabajo.

- El proceso de implementación estuvo guiado por la utilización de las tecnologías seleccionadas, logrando al final de las iteraciones pactadas y un producto con las funcionalidades requeridas.

- Para la validación del *software* se realizaron pruebas unitarias y de aceptación al sistema, planteadas por la metodología XP. Se detectaron un total de 31 No

Conformidades en tres iteraciones las cuales fueron resueltas al finalizar cada iteración, mejorando así la calidad del sistema.

Conclusiones generales

- El estudio de las soluciones similares existentes a nivel nacional y foráneo evidenció que no existe un sistema que garantice la informatización de los procesos de la actividad del juez de ejecución en Cuba. Demostrándose la necesidad de crear un sistema informático para la gestión de la información de estos procesos en el sistema de tribunales cubanos.

- El modelo conceptual permitió un mayor entendimiento. Además, se realizó el levantamiento de requisitos funcionales y no funcionales. Se obtuvo el diagrama de clases del diseño y el modelo entidad relación.

- Para la construcción de un sistema informático guiado por la metodología XP se usó como herramientas y tecnologías Django, UML, Visual Paradigm, Python, JavaScript, HTML 5 y CSS 3, jQuery y Bootstrap, JetBrains PyCharm.

- El desarrollo dirigido por pruebas que define la metodología seleccionada mediante pruebas unitarias y pruebas de aceptación permitió un producto final con la calidad requerida. Se utilizó la biblioteca ***TestCase*** que provee el *framework* Django para las pruebas unitarias, en el caso de las pruebas de aceptación se realizaron en conjunto con el cliente.

Recomendaciones

Para garantizar la continuidad de este trabajo y el enriquecimiento del mismo se recomienda:

- Integrarlo con los sistemas existentes en el sistema de tribunales cubanos.
- Incrementar al mismo las funcionalidades de la sala de ejecución.
- Aplicar en la medida de lo posible técnicas de inteligencia artificial a la información que aporta el sistema.

Referencias

Abreu, P. E. A. (2019). *Tema: Mantenimiento al sistema de gestión de pacientes del Centro Oncológico del Hospital General Universitario Vladimir Ilich Lenin*. Universidad de Holguín.

Acosta, M. P., & Martínez, C. M. de L. (2015). *Sistema informático de apoyo a los procesos " Evaluación del estudiante " y " Censo , capacidad y matrícula " en la Direcció n de la Residencia Estudiantil 2 de la Universidad de las Ciencias Informática*s. Universidad de las Ciencias Informáticas.

Alba, R. H. P. (2018). *Automatización de procesos para planificación curricular e incidencia en labor docente de la Unidad Educativa Teodoro Gómez de la Torre, utilizando metodología Extreme Programming*. UNIVERSIDAD TÉCNICA DEL NORTE, Ecuador.

Alegsa, L. (2017). *Definiciones*. https://www.definiciones-de.com

Anwer, F., Aftab, S., Waheed, U., & Muhammad, S. S. (2017). Agile Software Development Models TDD , FDD , DSDM , and Crystal Methods : A Survey. *INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY SCIENCES AND ENGINEERING, 8*. www.ijmse.org

Batista, A. R. (2005). Impacto social de la ciencia y la tecnología en Cuba : una experiencia de medición a nivel macro. *Ciencia Tecnología y Sociedad*, *2*, 47–171.

Boaventura, J., Herrera, E. P., Vicet, P. V., & Alvarez, Y. F. (2016). Elección entre una metodología ágil y tradicional basado en técnicas de soft computing. *Revista Cubana de Ciencias Informáticas*, *10*, 145–158.

Boehm, B., & Turner, R. (2003). *Balancing Agility and Discipline: AGuide for the Perplexed.* (I. Pearson Education (ed.)).

Bruzón, F. A. C. (2019). *Módulo de gestión de datos de biopsias para el registro de cáncer en el centro Oncológico de Holguín*. Universidad de Holguín.

Caccia, E. (2019). *Arquitectura MVC - Parte 1*. https://medium.com/somos-codeicus/arquitectura-mvc-conceptos-básicos-481062755df9

Costa, H. (2019). *Patrón MVT : Modelo Vista Template*. https://github.com/Gohanckz/workshop-django/issues/19

Demestre, D. H., & González, Y. H. (2019). Diseño de un Sistema de Gestión de Información de Recursos Humanos. *Serie Científica de La Universidad de Las Ciencias Informáticas*, *12*(3), 31–40.

DesarrolloWeb.com. (2003). *Qué es Python*. https://desarrolloweb.com/articulos/1325.php

Di Pierro, M. (2014). Capitulo II. Que es un framework web2py y que tecnología usa. In T. L. Editions (Ed.), *web2py (5th Edition)*.

Enríquez Ruiz, J. L., Farías Palacín, E., Flores Flores, E., & Honores Solano, C. (2017). *METODOLOGÍA DE DESARROLLO DE SOFTWARE* (pp. 1–39).

Ferro, R. R. (2020). *Sistema de Tribunales en Cuba, en busca de más calidad en la justicia que se imparte*. https://www.tsp.gob.cu/noticias/sistema-de-tribunales-en-cuba-en-busca-de-mas-calidad-en-la-justicia-que-se-imparte

GetApp. (2018a). *eCourt*. https://www.getapp.es/software/103805/ecourt

GetApp. (2018b). *Matrix*. https://www.getapp.es/software/9268/matrix

GetApp. (2019a). *iJustice*. https://www.getapp.es/software/9299/ijustice

GetApp. (2019b). *Noble Justice*. https://www.getapp.com.co/software/2049509/noble-justice

GMI Blogger. (2021). *20 Best Frontend & Backend Web Development Frameworks 2022*. September, 2nd. https://www.globalmediainsight.com/blog/web-development-frameworks/

Jansen, A., Malavolta, I., Muccini, H., Ozkaya, I., & Goos, G. (2020). *Software Architecture*.

Jaramillo, C. A. B. (2015). TDD - TEST DRIVEN DEVELOPMENT METODOLOGIAS AGILES Y CIRCULO DE. *ResearchGate*, *December*. https://doi.org/10.13140/RG.2.1.1911.0483

JetBrains. (2020). *The Python IDE for Professional Developers*.

Labañino Urbina, S., Toledano López, O. G., & Labañino Gainza, C. (2020). Algoritmo de búsqueda fonética para el Sistema de Gestión Integral de Aduanas. *Serie Científica de La Universidad de Las Ciencias Informáticas*, *13*(9), 82–96.

Larman, C. (2003). *UML y Patrones*.

Maida, E., & Pacienzia, J. (2015). *METODOLOGIAS DE DESARROLLO DE SOFTWARE*. Universidad Católica Argentina.

MDM, W. D. (2019). *Introducción a Django*. https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introduction

Peñaherrera, O. M. (2021). *DESARROLLO DE UNA APLICACIÓN WEB PARA APRENDIZAJE BÁSICO DE LENGUA DE SEÑAS ECUATORIANO*. ESCUELA POLITÉCNICA NACIONAL DE PERÚ.

Pilataxi Alba, E. R. (2018). *"E-portafolio y su incidencia en los procesos de evaluación de docentes en la Unidad Educativa Ibarra, utilizando la metodología Extreme Programming."* UNIVERSIDAD TÉCNICA DEL NORTE INSTITUTO, Ecuador.

Postgres. (2021). *¿Que es PostgreSQL?* www.postgresql.org

PostgreSQL. (2021). *¿Por qué utilizar PostgreSQL?* www.postgresql.org

Presidencia de Cuba,. (2019). *Programas priorizados por el gobierno cubano*. https://www.presidencia.gob.cu/es/gobierno/programas-priorizados/

Quality Devs. (2019). *¿Qué es Symfony?* https://www.qualitydevs.com/2019/08/05/que-es-symfony/

RAE. (2020). *Diccionario de la Lengua Española*. https://dle.rae.es

Reitz, K. (2018). *Python Guide Documentation*. https://buildmedia.readthedocs.org/media/pdf/python-guide/latest/python-guide.pdf

Rodríguez Sánchez, T. (2014). *Metodología de desarrollo para la Actividad productiva de la UCI .*

Roman, A., & Mnich, M. (2021). Test - driven development with mutation testing – an experimental study. In *Software Quality Journal*. Springer US. https://doi.org/10.1007/s11219-020-09534-x

Romero, Y. F., & González, Y. D. (2012). Patrón Modelo-Vista-Controlador. *Revista Digital de Las Tecnologías de La Información y Las Comunicaciones*, *11*(1), 47–57.

Rumbaugh, J., Jacobson, I., & Booch, G. (2000). *El Lenguaje Unificado de Modelado.*

Saiedian, H. (1997). Una evaluación del del modelo entidad relación extendido. *Information and Software Technology*, *39*, 449–462. https://users.dcc.uchile.cl/~cgutierr/cursos/BD/extendido.pdf

Schwaber, K., & Sutherland, J. (2017). *La Guía de Scrum*.

Sohaib, O., Solanki, H., Dhaliwa, N., Hussain, W., & Asif, M. (2018). Integrating design thinking into extreme programming. *Journal of Ambient Intelligence and Humanized Computing*. https://doi.org/10.1007/s12652-018-0932-y

Toledano, O. G., & Camps, E. B. (2015). *Portal del Centro de Recursos para el Aprendizaje y la Investigación de la Facultad de Economía de la Universidad de La Habana*. Universidad de las Ciencias Informáticas.

Toro, A., & Peláez, L. E. (2016). Ingeniería de Requisitos: de la especifiación de requisitos de software al aseguramiento de la calidad. Cómo lo hacen las Mipymes desarrolladoras de software de la ciudad de Pereira. *Entre Ciencia e Ingeniería*, *20*, 117–123.

Tribunal Supremo Popular. (2019). *Organización del Sistema de Tribunales Cuba*. Organización Del Sistema de Tribunales Cuba. https://www.tsp.gob.cu/organizacion-del-sistema-de-tribunales

Tribunal Supremo Popular. (2020). *Estrategia para el desarrollo científico-tecnológico y la innovación en el Sistema de Tribunales*. https://www.tsp.gob.cu/sites/default/files/documentos/Estrategia para el desarrollo científico-tecnológico y la innovación en el Sistema de Tribunales.pdf

Venters, C. C., Capilla, R., Betz, S., Penzenstadler, B., Crick, T., Crouch, S., Nakagawa, E. Y., Becker, C., & Carrillo, C. (2017). Software Sustainability: Research and Practice from a Software Architecture Viewpoint. *The Journal of Systems & Software*. https://doi.org/10.1016/j.jss.2017.12.026

Vera, R. A. (2021). *Qué es Laravel: Características y ventajas*. https://openwebinars.net/blog/que-es-laravel-caracteristicas-y-ventajas/

Villa, B. (2019). *Estudio Comparativo Sobre Las Metodologías Open Up (Open Unified Process) Y Xp (Extreme Programming) Como Modelos Ágiles Para El Desarrollo De Software*. 29.

Visual Paradigm. (2019). *Visual Paradigm.Who We Are?* https://visual-paradigm.com/

Anexos

Anexo 1: Historias de usuario

Tabla 24: HU1- Gestionar usuario

Historia de usuario	
Número: 1	**Nombre:** Gestionar usuario
Usuarios: Administrador	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Estimación: 3 días	**Iteración Asignada:** 1
Descripción: Los administradores podrán gestionar los usuarios del sistema, así como el rol que desempeñan.	
Observaciones: Incluye adicionar, listar, modificar y eliminar usuario.	

Tabla 25: HU5- Gestionar tribunal

Historia de usuario	
Número: 5	**Nombre:** Gestionar tribunal
Usuarios: Administrador	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Estimación: 2 días	**Iteración Asignada:** 1
Descripción: El administrador gestiona todo lo referente a los tribunales.	
Observaciones: El usuario se encargará de adicionar, listar, modificar y eliminar tribunales.	

Tabla 26: HU6- Gestionar provincia

Historia de usuario	
Número: 6	**Nombre:** Gestionar provincia
Usuarios: Administrador	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Estimación: 3 días	**Iteración Asignada:** 1
Descripción: El administrador gestiona todo lo referente a las provincias.	
Observaciones: El usuario se encargará de adicionar, listar, modificar y eliminar provincias.	

Tabla 27: HU7- Gestionar municipio

Historia de usuario	
Número: 7	**Nombre:** Gestionar municipio
Usuarios: Administrador	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Estimación: 3 días	**Iteración Asignada:** 1
Descripción: El administrador gestiona todo lo referente a los municipios.	
Observaciones: El usuario se encargará de adicionar, listar, modificar y eliminar municipios.	

Tabla 28: HU9- Gestionar consejo popular

Historia de usuario	
Número: 9	**Nombre:** Gestionar consejo popular
Usuarios: Administrador	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Estimación: 3 días	**Iteración Asignada:** 1
Descripción: El administrador será el encargado de la gestión de lo referente a los consejos populares.	
Observaciones: El usuario podrá añadir, listar y modificar los traslados de los sancionados.	

Tabla 29: HU10- Gestionar radicado sin documento

Historia de usuario	
Número: 10	**Nombre:** Gestionar radicado sin documento
Usuarios: Asistente	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Estimación: 3 días	**Iteración Asignada:** 2
Descripción: El asistente será encardado de añadir al sistema a los radicados sin documentos que posteriormente se convertirán en sancionados.	
Observaciones: Al pasar a formar parte de los sancionados, estos son eliminados de la lista de radicados sin documentos.	

Tabla 30: HU11- Gestionar visita

Historia de usuario	
Número: 11	**Nombre:** Gestionar visita

Usuarios: Asistente	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Estimación: 3 días	**Iteración Asignada:** 2
Descripción: El asistente registrara la visita a un sancionado, cuando un asistente de ejecución le entregue los datos de la visita.	
Observaciones: El usuario podrá añadir, listar, modificar las visitas. Una vez que el sancionado cause baja del sistema se eliminan las visitas que se le realizaron.	

Tabla 31: HU12- Gestionar asistente de ejecución

Historia de usuario	
Número: 12	**Nombre:** Gestionar asistente de ejecución
Usuarios: Administrador	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Estimación: 3 días	**Iteración Asignada:** 1
Descripción: El administrador se encarga de todas las tareas de gestión del asistente de ejecución.	
Observaciones: El usuario podrá añadir, listar, modificar y eliminar asistentes de ejecución.	

Anexo 2: Tareas de Ingeniería

Tabla 32: Tarea de ingeniería 5. Adicionar delito

Tarea	
Número de Tarea: 5	**Número de la Historia de usuario: 3**
Nombre de la Tarea: Adicionar Delito	
Tipo de Tarea: Desarrollo	**Estimación:** 1 día
Fecha de inicio: 20-10-21	**Fecha de fin:** 20-10-21
Programador responsable: Hugo Alberto Valencia Zayas	
Descripción: Implementar una vista que permita adicionar un delito al sistema.	

Tabla 33: Tarea de ingeniería 6. Modificar delito

Tarea	
Número de Tarea: 6	**Número de la Historia de usuario: 3**
Nombre de la Tarea: Modificar Delito	

Tipo de Tarea: Desarrollo	**Estimación:** 1 día
Fecha de inicio: 20-10-21	**Fecha de fin:**
Programador responsable: Hugo Alberto Valencia Zayas	
Descripción: Implementar una vista que permita modificar un delito del sistema.	

Tabla 34: Tarea de ingeniería 7. Listar delito

Tarea	
Número de Tarea: 7	**Numero de la Historia de Usuario: 3**
Nombre de la Tarea: Listar Delito	
Tipo de Tarea: Desarrollo	**Estimación:** 1 día
Fecha de inicio: 21-10-21	**Fecha de fin:** 21-10-21
Programador responsable: Hugo Alberto Valencia Zayas	
Descripción: Implementar una vista que permita listar los delitos del sistema	

Tabla 35: Tarea de ingeniería 8. Eliminar delito

Tarea	
Número de Tarea: 8	**Número de la Historia de usuario: 3**
Nombre de la Tarea: Eliminar Delito	
Tipo de Tarea: Desarrollo	**Estimación:** 1 día
Fecha de inicio: 22-10-21	**Fecha de fin:** 22-10-21
Programador responsable: Hugo Alberto Valencia Zayas	
Descripción: Implementar una vista que permita eliminar un usuario al sistema.	

Tabla 36: Tarea de ingeniería 9. Adicionar sanción

Tarea	
Número de Tarea: 9	**Número de la Historia de usuario: 4**
Nombre de la Tarea: Adicionar Sanción	
Tipo de Tarea: Desarrollo	**Estimación:** 1 día
Fecha de inicio: 25-10-21	**Fecha de fin:** 25-10-21
Programador responsable: Hugo Alberto Valencia Zayas	
Descripción: Implementar una vista que permita adicionar una sanción al sistema.	

Tabla 37: Tarea de ingeniería 10. Modificar sanción

Tarea	
Número de Tarea: 10	**Número de la Historia de usuario: 4**
Nombre de la Tarea: Modificar Sanción	
Tipo de Tarea: Desarrollo	**Estimación:** 1 día
Fecha de inicio: 26-10-21	**Fecha de fin:** 26-10-21
Programador responsable: Hugo Alberto Valencia Zayas	
Descripción: Implementar una vista que permita modificar una sanción al sistema.	

Tabla 38: Tarea de ingeniería 11. Listar sanción

Tarea	
Número de Tarea: 11	**Número de la Historia de usuario: 4**
Nombre de la Tarea: Listar Sanción	
Tipo de Tarea: Desarrollo	**Estimación:** 1 día
Fecha de inicio: 27-10-21	**Fecha de fin:** 27-10-21
Programador responsable: Hugo Alberto Valencia Zayas	
Descripción: Implementar una vista que permita listar las sanciones del sistema.	

Tabla 39: Tarea de ingeniería 12. Eliminar sanción

Tarea	
Número de Tarea: 12	**Número de la Historia de usuario: 4**
Nombre de la Tarea: Eliminar Sanción	
Tipo de Tarea: Desarrollo	**Estimación:** 1 día
Fecha de inicio: 28-10-21	**Fecha de fin:** 28-10-21
Programador responsable: Hugo Alberto Valencia Zayas	
Descripción: Implementar una vista que permita eliminar una sanción al sistema.	