Universidad de las Ciencias Informáticas
Vertex, Facultad de Tecnologias Interactivas

Sistema automatizado para la gestión y monitoreo del bombeo de agua en la Universidad de las Ciencias Informáticas

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Carlos Adrián Burley Ross

Tutor: Ing. Julio Alberto Leyva Durán

La Habana, 2024

Aquellos que pueden imaginar cualquier cosa, pueden crear lo imposible
Alan Turing

Dedicatoria

Dedico este trabajo a mi novia, por su amor incondicional y su fortaleza en cada adversidad; a mi madre, por su apoyo incansable y su fe en mí; a mi padre, por su guía y respaldo; y a toda mi familia, por su presencia y ayuda en los momentos más difíciles. Sin ustedes, este logro no habría sido posible.

Agradecimientos

Quiero expresar mi más profundo agradecimiento a todas las personas que han sido parte esencial de este camino y que, de una u otra forma, contribuyeron a que este proyecto de tesis se hiciera realidad.

En primer lugar, a mi novia, quien ha estado a mi lado incondicionalmente, enfrentando conmigo cada adversidad a pesar de sus propias dificultades. Su fortaleza, paciencia y amor han sido una fuente constante de motivación y apoyo, a lo que siempre estaré agradecido.

A mi madre, por ser un pilar inquebrantable de apoyo y cariño en todo momento. Su dedicación, sacrificio y fe incondicional en mí han sido el motor que me impulsó a seguir adelante, incluso en los momentos más difíciles. A mi padre, por su guía y respaldo, y a mi familia en general, por estar presentes en cada etapa de este proceso, ofreciendo su ayuda y compañía cuando más la necesitaba.

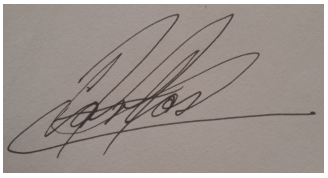
No puedo dejar de mencionar a mis compañeros de grupo, especialmente a Alejandro Santana, Alejandro Labaut, Javier y Carlos Bryan, quienes no solo compartieron este recorrido, sino que también me brindaron su asistencia y ayuda en más de una ocasión. Su compromiso y compañerismo han sido significativos en el proceso y sobre todo en los resultadosA mi tutor de tesis, gracias por su guía y en cada etapa de este trabajo

A todos ustedes, mi más sincero agradecimiento. Este logro no es solo mío, sino también de quienes me acompañaron, apoyaron y creyeron en mí a lo largo de este camino.

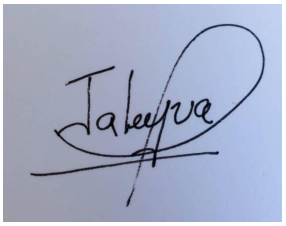
Declaración de autoría

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

Para que así conste firmamos la presente a los **10** días del mes de **diciembre** del año **2024**.



Carlos Adrián Burley Ross
Autor



Ing. Julio Alberto Leyva Durán
Tutor

Resumen

La Universidad de las Ciencias Informáticas ha enfrentado diversos desafíos relacionados con la gestión del bombeo de agua en sus instalaciones desde su fundación. Este proceso, tradicionalmente manual, ha generado ineficiencias en la administración del recurso hídrico, lo que motivó el desarrollo de esta investigación para su informatización. El trabajo investigativo identifica y describe los procesos vinculados a la gestión del bombeo de agua, con énfasis en aquellos que impactan directamente en la eficiencia del uso y distribución del agua en la universidad. El análisis realizado evidencia la necesidad de implementar un sistema que satisfaga los requerimientos específicos para la gestión del recurso hídrico. El desarrollo estuvo guiado por la metodología Programación extrema (XP, por sus siglas en inglés) y se seleccionó como principales tecnologías,el protcolo de comunicación Transporte de telemetría de Message Queue Server (MQTT, por sus siglas en inglés) el framework Django de Python y SQLite3 como Sistema Gestor de Base de Datos (SGBD). Para verificar la calidad y funcionalidad del sistema, así como el ajuste a las expectativas del cliente se realizaron pruebas de rendimiento, unitarias y de aceptación. El Sistema automatizado desarrollado mejora la eficiencia y efectividad en la gestión del agua en la universidad.

Palabras clave: Bombeo, Desarrollo, Eficiencia, Gestión, SCADA.

Índice general

Índice de figuras

Índice de tablas

Introducción

La automatización de procesos industriales y de gestión de recursos ha evolucionado significativamente con el desarrollo de las tecnologías digitales. Los sistemas de control y supervisión modernos permiten monitorear variables ambientales y optimizar el uso de recursos energéticos en tiempo real, facilitando la gestión eficiente de instalaciones (García Ruiz, 2021).

Los Supervisión, Control y Adquisición de Datos (SCADA, por sus siglas en inglés) constituyen una herramienta fundamental en la automatización industrial. Estos sistemas integran la comunicación con dispositivos automáticos programados y facilitan el procesamiento de datos desde ubicaciones remotas (R. A. M. J. y C., 2017). En el contexto cubano, el desarrollo de SCADA ha demostrado su efectividad, como evidencia la implementación del Sistema EROS en la empresa niquelífera Ernesto Che Guevara en 1993, que optimizó significativamente sus procesos industriales (M. R., 2022).

La UCI, a través de su Centro de Tecnologías Interactivas, ha contribuido al desarrollo de soluciones SCADA. El sistema Guardian del alba (GALBA), implementado exitosamente en la empresa PDVESA, y el desarrollo actual del mini SCADA Arex, demuestran la capacidad institucional en el desarrollo de sistemas de control y supervisión adaptados a necesidades específicas (al, 2020).

La evolución hacia la Industria 4.0 ha impulsado el desarrollo de plataformas de Internet de las cosas (Iot) para la automatización de infraestructuras. Estas plataformas facilitan la gestión energética, el control de variables ambientales y la optimización de recursos en diversos sectores, incluyendo instalaciones educativas y administrativas.

La gestión eficiente de portadores energéticos requiere del establecimiento de procedimientos de control y evaluación que cumplan con las regulaciones establecidas. Sin embargo, la UCI enfrenta actualmente limitaciones en el control y supervisión de estos portadores, particularmente en la gestión del bombeo de agua. Específicamente en la Estación de Bombeo de agua no se tiene un control de la rotación de los operadores que interactúan con el control de presión de agua hacia el campus universitario, no posee un registro sobre los salideros y reparaciones de las tuberías afectadas por las altas presiones discontinuas, la planificación de la presión del agua no se controla de forma automática y no se difunde a tiempo las afectaciones por los canales formales de la universidad.

En este sentido, se señala como insuficiencia el control de los portadores energéticos en la UCI, específicamente en el bombeo de agua, lo cual conspira contra el funcionamiento adecuado del campus universitario. Asimismo, la lejanía de la Universidad de la Estación de Bombeo, la situación económica actual y la necesidad de proteger los bienes que posee la Universidad, constituye una tarea de primer orden. Se impone así, la búsqueda de una solución tecnológica que permita monitorear oportunamente el orden de la distribución de agua de la Universidad.

Teniendo en cuenta la situación problémica antes descrita, se identifica como problema de investigación: ¿Cómo garantizar el monitoreo y gestión de la distribución de agua de la Universidad?

Como objeto de estudio se define los procesos de gestión y monitoreo automático de bombeo de agua.

Como campo de acción: Sistema de gestión para la distribución automática del bombeo de agua en la UCI.

Con el fin de solucionar el problema planteado se define como objetivo general: Desarrollar un Sistema de gestión web para el monitoreo y gestión del bombeo de agua en la UCI.

Posibles resultados: Sistema web para el monitoreo y gestión del bombeo de agua en la UCI.

Planificación de la investigación:

- Elaboración de los fundamentos teórico - metodologicos de los procesos de monitoreo y gestión del bombeo de agua.
- Generación de los artefactos relacionados con el análisis y diseño de la solución propuesta
- Implementación de la solución propuesta.
- Validación de las principales funcionalidades a través de pruebas de software..

Para el desarrollo de esta investigación se emplearon los siguientes métodos científicos:
Métodos teóricos:

Modelación: Se aplica para representar mediante diagramas UML las características, procesos y componentes del sistema de gestión de bombeo de agua, permitiendo visualizar las relaciones entre sus elementos y la arquitectura de la solución.
Analítico-sintético: Se utiliza durante la revisión de la literatura especializada para identificar las características fundamentales de los sistemas SCADA y las plataformas IoT, así como para determinar los elementos esenciales que debe incluir el sistema de gestión propuesto.

Métodos empíricos:

Entrevista: Se realizan entrevistas semiestructuradas al personal de la dirección energética de la UCI y a los operadores de la estación de bombeo para identificar los requisitos funcionales del sistema y comprender los procesos actuales de gestión del agua.
Observación: Se emplea para analizar el funcionamiento actual de la estación de bombeo y documentar los procesos manuales que requieren automatización.

CAPÍTULO 1

FUNDAMENTOS Y REFERENTES TEÓRICO-METODOLÓGICOS SOBRE EL OBJETO DE ESTUDIO

1.1. Introducción

En este capítulo se exponen los fundamentos teóricos relacionados con el trabajo de investigación, necesarios para el correcto entendimiento de la solución propuesta, incluyendo la definición, objetivos y requisitos de la gestión y monitoreo del bombeo de agua. Se analizarán los principales conceptos relacionados con el tema de investigación, así como las características fundamentales de las herramientas y tecnologías para el desarrollo.

1.2. Sistema de gestión de bombeo de agua.

La gestión de portadores energéticos constituye un aspecto fundamental para el desarrollo eficiente de cualquier institución, ya que permite optimizar el uso de recursos esenciales y contribuir a la sostenibilidad económica y ambiental. En este contexto, las instituciones educativas como la UCI gestionan diversos portadores energéticos, entre los que se destacan el gas, el combustible y el agua, elementos críticos para el funcionamiento cotidiano de la institución(B. y A. C. A., 2022).

Particularmente, en lo que respecta al agua, la universidad actualmente enfrenta limitaciones significativas en su gestión, realizando los procesos de control y monitoreo de manera manual. Esta situación evidencia la necesidad de implementar soluciones tecnológicas que permitan una administración más eficiente y precisa del recurso hídrico.

El proceso de bombeo de agua representa un sistema complejo que involucra múltiples componentes técnicos y operativos. En la UCI, este proceso se desarrolla a través de una estación de bombeo perteneciente a la

empresa Aguas de La Habana, ubicada en un punto estratégico. La infraestructura consta de un tanque principal que recibe agua desde el Rincón, una zona aledaña de la capital, y que cuenta con una configuración hidráulica específica(G. B. R. M. S. P. y R. A., 2022).

La estación de bombeo presenta una arquitectura caracterizada por tres tuberías, cada una equipada con una turbina eléctrica que facilita el transporte del agua. Estas turbinas se encuentran conectadas a un variador de frecuencia, lo cual permite un control más preciso del flujo. Posteriormente, las tres tuberías confluyen en una única línea que conduce el agua hacia el campus universitario, configurando un sistema integral de distribución hídrica(P. S. M. S. R. F. J. R. H. M. y L. J. P. A., 2020).

1.2.1. Proceso de Medición de Datos

En la actualidad, el proceso de medición de datos en la estación de bombeo de la Universidad de Ciencias Informáticas se caracteriza por una metodología principalmente manual, que involucra diversos instrumentos de control y registro.

El sistema de monitoreo cuenta con dos relojes estratégicamente ubicados en el punto de convergencia de las tres tuberías que emergen del tanque principal. Uno de estos instrumentos está destinado a medir la presión de salida del agua que se bombea hacia el campus universitario, mientras que el otro registra el volumen de agua bombeado, proporcionando información crucial sobre el flujo hídrico.Un elemento central en este proceso de control es el Variador de Frecuencia Variador de Frecuencia (VFD, por sus siglas en inglés), conectado a los tres motores. Este dispositivo desempeña un papel fundamental en la regulación de la presión de salida, con la capacidad de regular la velocidad de los motores de manera automática en función de los parámetros de presión configurados.

Adicionalmente, se ubica otro reloj de medición en la tubería que conduce el agua desde el Rincón hacia el tanque, cuya función es monitorear la presión de entrada. En cuanto al nivel del tanque, anteriormente se utilizaba un sensor para su medición, pero actualmente esta tarea se realiza mediante una apreciación visual directa.La recopilación de estos datos se efectúa de manera sistemática, con un registro cada hora que se consigna manualmente en una planilla operacional. Este método, aunque garantiza un seguimiento continuo, presenta limitaciones inherentes a los procesos manuales, como la posibilidad de errores humanos y la falta de un registro en tiempo real.

La descripción detallada de este proceso de medición evidencia la necesidad de implementar sistemas automatizados que permitan una monitorización más precisa, eficiente y continua de los parámetros críticos en la estación de bombeo.

1.2.2. PLC Industrial basada en hardware de código abierto

Este PLC basado en Arduino, como se ilustra en la figura 1.1, está especialmente diseñado para su uso en un entorno industrial. Este autómata dispone de varias Entradas/Salidas (E/S), también dispone de diferentes interfaces de comunicación lo que le ofrece una gran flexibilidad y control. El PLC IOC ofrece la posibilidad de expandirse con otros módulos mediante el sistema I2C, lo que significa que puede gobernar varias E/S en modo maestro esclavo, además de módulos adicionales de sensores.

Figura 1.1. PLC Basado en Arduino https://www.industrialshields.com/es_ES

Software de programación

Este PLC industrial se puede programar utilizando la plataforma Arduino IDE y otros lenguajes o interfaces de programación, ya que es de código abierto.

Tratamiento de agua con PLC Industrial Arduino

El controlador Arduino se puede conectar a todos los sensores para detectar todos los indicadores en las instalaciones de bombeo de agua. Los controladores basados en Arduino y Raspberry Pi y los Panel PC

permiten conectar sensores analógicos para medir el pH, la clorina, la turbidez y otros factores clave que determinan la calidad del agua. Además es posible conectar sensores de nivel de tanque, digital o analógico, para garantizar que haya productos químicos suficientes para controlar las propiedades de calidad del agua bombeada hacia el campus universitario.
Mediante las salidas analógicas o con el encendido o apagado de las bombas, a través de un dispositivo actuador Variador de Frecuencia (o siglas VFD, del inglés: Variable Frequency Drive) se puede regular la velocidad de las bombas, lo que garantiza un control de la presión del agua asegurando los niveles de aprovechamiento del agua. Actualmente las variables que se miden en la bomba de agua de la UCI son: Presión de Entrada y Salida, El nivel de llenado del tanque, el volumen bombeado.

1.3. Sistemas Homólogos

En el campo de la gestión y bombeo de agua, existen diversos sistemas homólogos que han sido implementados tanto a nivel nacional como internacional. Estos sistemas comparten el objetivo común de optimizar el uso del recurso hídrico y mejorar la eficiencia energética en el proceso de bombeo.

En el contexto internacional, destaca el sistema WaterGEMS, de Bentley Systems, que integra análisis hidráulico y de calidad del agua con herramientas de gestión de activos. Switnicka; Suchorab y Kowalska, 2017 reportan que WaterGEMS ha sido utilizado con éxito en varias universidades estadounidenses para optimizar sus sistemas de distribución de agua, logrando ahorros significativos en costos operativos. Un aspecto reutilizable de este sistema es su capacidad de modelado hidráulico integrado, que permite simular diferentes escenarios de distribución de agua y predecir potenciales ineficiencias antes de implementar cambios reales.

En el ámbito latinoamericano, el sistema desarrollado por la Universidad Nacional Autónoma de México (UNAM) para la gestión de su red de agua potable es un referente importante. Según (G. B. R. M. S. P. y R. A., 2022), este sistema ha permitido una reducción del 15 porciento en el consumo de agua del campus universitario, además de mejorar la detección temprana de fugas. Este sistema aporta a la solución su diseño de interfaz minimalista con gráficos de consumo de fácil comprensión, que permite a usuarios no técnicos interpretar rápidamente la información.

En Cuba, el desarrollo de sistemas de gestión y bombeo de agua ha estado limitado por factores económicos y tecnológicos. Sin embargo, existen iniciativas destacables como el sistema SCADA GALBA, desarrollado en la propia UCI. Aunque GALBA fue concebido originalmente para la industria petrolera, sus principios de automatización, supervisión y control remoto son igualmente aplicables a la gestión del agua, permitiendo monitorear y operar procesos a distancia, optimizar recursos y garantizar la continuidad del servicio (Rodriguez Capote; Hernandez Barriento; Corrales Diaz; Gomez Garcia y Fernandez Infante, 2016).

Por otro lado, el Instituto Superior Politécnico José Antonio Echeverría (ISPJAE) ha implementado un sistema de monitoreo y control para la red de abastecimiento de agua en La Habana, reportado por Raimundo Carlos, 2019. Este sistema, aunque inicialmente no diseñado para entornos universitarios, aporta valiosas lecciones sobre cómo adaptar tecnologías de gestión del agua a las condiciones específicas del contexto cubano, tales como la integración de infraestructuras existentes, la optimización del uso energético y la priorización de zonas críticas.

Un aspecto común en estos sistemas es la integración de tecnologías Iot para la recolección de datos en tiempo real. DusunIoT, 2024 destacan la importancia de esta integración para mejorar la eficiencia energética y la gestión de recursos hídricos en sistemas de bombeo. La implementación de algoritmos de aprendizaje automático para la predicción de demanda y detección de anomalías es otra tendencia creciente en estos sistemas. Larrañaga, 2024 demuestran cómo estos algoritmos pueden mejorar significativamente la eficiencia operativa de los sistemas de bombeo de agua.

A continuación, se presenta una tabla comparativa de los sistemas homólogos mencionados:

Tabla 1.1. Tabla comparativa entre homólogos

Sistema	**Origen**	**Contexto de Aplicación**	**Características Principales**	**Beneficios Reportados**
WaterGEMS	Internacional (Bentley Systems)	Universidades estadounidenses	Integración de análisis hidráulico y gestión de activos	Optimización de sistemas de distribución, ahorro en costos operativos
SCADA UNAM	México	Campus universitario	Control y monitoreo en tiempo real	Reducción del consumo de agua (15 %), detección temprana de fugas
SCADA GALBA	Cuba (UCI)	Industria petrolera (adaptable)	Automatización y control remoto	Aplicabilidad a gestión de agua (potencial)
Sistema ISPJAE	Cuba	Red de abastecimiento urbana	Monitoreo y control adaptado al contexto cubano	Mejora en la gestión de recursos hídricos en condiciones locales

Del análisis de los sistemas homólogos se identifican componentes informáticos fundamentales para el desarrollo de la solución propuesta:

La arquitectura cliente-servidor para el procesamiento distribuido, el uso de tecnologías web para la inter-

faz de usuario, y la implementación de protocolos de comunicación para la integración con sensores. Estos elementos técnicos, combinados con frameworks de desarrollo ágil y sistemas de gestión de bases de datos, proporcionan la base tecnológica para implementar una solución que responda a los requerimientos específicos de la UCI.
Desde la perspectiva del desarrollo de software, los sistemas analizados aportan patrones de diseño y estructuras de datos que pueden adaptarse al contexto local, permitiendo crear una solución escalable que integre el monitoreo en tiempo real con la gestión administrativa del bombeo de agua.

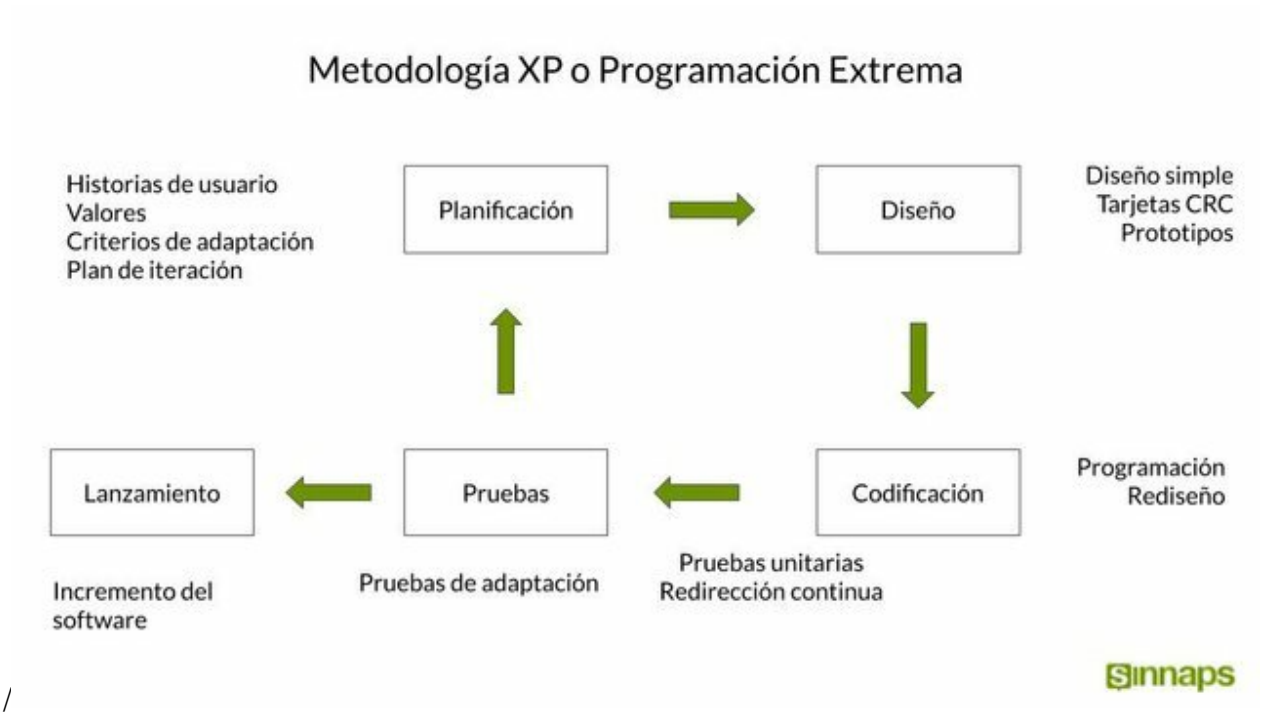
1.4. Metodologías de desarrollo

Las metodologías de desarrollo de software constituyen marcos de trabajo que establecen principios, prácticas y procedimientos para guiar el proceso de construcción de aplicaciones informáticas. Estas metodologías proporcionan una estructura sistemática que facilita la planificación, desarrollo y mantenimiento del software (Sommerville, 2011). La selección de una metodología adecuada resulta fundamental para el éxito de un proyecto, pues determina cómo se organizará el trabajo y qué prácticas se implementarán durante el desarrollo.

En el panorama actual del desarrollo de software, las metodologías han evolucionado desde enfoques tradicionales hacia aproximaciones más ágiles y adaptativas. Mientras las metodologías tradicionales se caracterizan por su exhaustiva documentación y procesos rigurosos, las metodologías ágiles enfatizan la adaptabilidad y la entrega incremental de valor (K. S. S. P. y K. M., 2022). Esta evolución responde a la necesidad de adaptarse a entornos cada vez más dinámicos y requisitos cambiantes.

El ciclo de desarrollo en XP se divide en iteraciones. Cada iteración incluye fases de planificación, diseño, codificación, pruebas, integración y entrega como se refleja en la figura 1.4. A través de estas iteraciones, el software se construye de manera incremental, permitiendo obtener retroalimentación del cliente en cada etapa del proceso y realizar los ajustes necesarios (B. K. y Andres, 2004). Además, XP utiliza varios artefactos que son clave para la organización y ejecución del proyecto, entre ellos se destacan:

- Historias de Usuario: Describen las funcionalidades desde la perspectiva del usuario.
- Tarjetas de Tareas: Detallan las acciones específicas que deben completarse.
- Pruebas unitarias y de aceptación: Se emplean para asegurar la calidad del software, realizando verificaciones automáticas y manuales que garantizan el cumplimiento de los requisitos del cliente (C. M., 2004).



/

Figura 1.2. Metodología XP, Programación Extrema. DiegoCalvo

Según un estudio realizado por R. M. G. L. y T. J., 2019 en el contexto nacional , la aplicación de XP en proyectos de software de mediana escala ha demostrado una mejora del (30 %) en la calidad del producto final y una reducción del (20 %) en los tiempos de entrega, en comparación con metodologías tradicionales. La metodología XP destaca entre las opciones ágiles por su enfoque en la calidad técnica y la adaptabilidad. S. K. y Sutherland, 2020 señalan que XP resulta particularmente efectiva en proyectos con equipos pequeños y requisitos evolutivos, características presentes en el desarrollo del sistema de gestión de bombeo de agua. La metodología prioriza la simplicidad, la comunicación y la retroalimentación continua, elementos esenciales para el desarrollo de sistemas críticos.

Para el desarrollo del sistema de gestión de bombeo de agua, XP emerge como la opción más apropiada debido a tres factores fundamentales: su capacidad para manejar requisitos cambiantes, su énfasis en la calidad del código, y su compatibilidad con el marco tecnológico seleccionado.(M. L. y R. R., 2023) demuestran que los proyectos que implementan XP en contextos similares logran mayor adaptabilidad y calidad en el producto final.

Su implementación en este proyecto facilita la integración continua de nuevas funcionalidades y la adaptación a los requisitos específicos de la UCI. (X. y C. J., 2023) argumentan que la metodología XP resulta especialmente efectiva en proyectos que requieren alta confiabilidad y adaptación constante a las necesidades del usuario, características esenciales en sistemas de control y monitoreo.

1.5. Herramientas y tecnologías para el desarrollo de la solución.

En el proceso de desarrollo de software, las herramientas juegan un papel fundamental, ya que permiten a los desarrolladores gestionar, codificar, depurar y mantener el sistema de manera eficiente (S. y M. B. R., 2015). Estas herramientas pueden abarcar desde entornos de desarrollo integrado (IDEs), hasta frameworks y lenguajes de programación específicos. El uso correcto de estas herramientas garantiza la creación de soluciones robustas y escalables, lo que resulta esencial para el éxito de proyectos informáticos complejos como la gestión automatizada del sistema de bombeo de agua.

1.5.1. Framework

Un framework es una estructura predefinida que facilita el desarrollo de aplicaciones, ya que proporciona un conjunto de herramientas y bibliotecas que permiten al programador enfocarse en la lógica específica del proyecto en lugar de las tareas repetitivas o complejas relacionadas con la construcción de una aplicación desde cero (G. D. S. D. J. y H., 2022). Los frameworks definen una arquitectura para el software, organizando el código y permitiendo la reutilización de componentes.

Existen diversos frameworks en el ámbito del desarrollo web, cada uno con características particulares que los hacen más o menos adecuados para ciertos tipos de proyectos. Algunos de los frameworks más utilizados en el desarrollo de aplicaciones web incluyen:

- Ruby on Rails: Popular por su enfoque en la simplicidad y productividad. Facilita el desarrollo rápido de aplicaciones web con convenciones sobre configuración, lo que permite a los desarrolladores trabajar de manera más eficiente.
- Laravel: Framework basado en PHP que es conocido por su facilidad de uso y su estructura robusta para desarrollar aplicaciones web complejas.
- Spring Boot: Utilizado para el desarrollo en Java, es uno de los frameworks más populares en el ecosistema empresarial por su capacidad para construir aplicaciones escalables y de alto rendimiento.
- Django: Framework de desarrollo web escrito en Python que sigue el principio de "no te repitas"(DRY) y fomenta la creación rápida de aplicaciones seguras y escalables.

En comparación con frameworks como Ruby on Rails o Laravel, Django se distingue por su enfoque en la simplicidad y seguridad desde el inicio. Rails ofrece una experiencia similar, pero Django se destaca por su comunidad sólida y su integración nativa con herramientas de administración. Mientras que Spring Boot es una excelente opción para aplicaciones empresariales en Java, su complejidad lo hace menos adecuado para un proyecto como el de la gestión de bombeo de agua, que puede beneficiarse más de la rapidez de desarrollo que ofrece Django y la facilidad de personalización en Python.

Entre todos los frameworks, Django ha sido seleccionado para el desarrollo de la aplicación web del sistema de bombeo de agua debido a sus numerosas ventajas. Django es un framework de código abierto escrito en Python que sigue el patrón Modelo-Vista-Plantilla (MVT) (Foundation, 2023). Sus principales ventajas incluyen:

- Rapidez y eficiencia: Django facilita el desarrollo rápido al proporcionar una amplia gama de herramientas integradas, como un ORM (Object-Relational Mapping) para la gestión de bases de datos, un sistema de autenticación y un administrador de contenido automático.
- Seguridad: Django incorpora por defecto medidas de seguridad para proteger contra ataques comunes como la inyección SQL y el cross-site scripting (XSS) (H. A. y K.-M. J., 2009).
- Este framework está diseñado para manejar grandes volúmenes de tráfico y datos, lo que lo convierte en una excelente opción para proyectos que pueden crecer en tamaño y complejidad.

1.5.2. Lenguaje de programación.

Un lenguaje de programación es una herramienta fundamental que permite a los programadores escribir instrucciones que una computadora puede entender y ejecutar (Sebesta, 2016). Los lenguajes varían en cuanto a su sintaxis, propósito y capacidades, y la elección de un lenguaje adecuado es crucial para el éxito de cualquier proyecto de software.

Algunos de los lenguajes más utilizados en el desarrollo de aplicaciones web incluyen:

- JavaScript: Lenguaje de programación interpretado y ampliamente utilizado en el desarrollo del lado del cliente. Es ideal para la creación de interfaces interactivas y dinámicas en aplicaciones web.
- Java: Un lenguaje orientado a objetos que destaca por su capacidad para crear aplicaciones robustas y escalables. Es común en entornos empresariales y desarrollo móvil (Android).
- PHP: Es un lenguaje de scripting diseñado para el desarrollo web. Su integración sencilla con HTML y bases de datos lo ha hecho popular, aunque tiene ciertas limitaciones en cuanto a rendimiento.
- Ruby: Utilizado con el framework Ruby on Rails, es conocido por su simplicidad y capacidad para facilitar el desarrollo rápido de aplicaciones.
- Python: Un lenguaje de programación de alto nivel, muy versátil y popular debido a su sintaxis sencilla, legible y su enfoque en la eficiencia del código.

El lenguaje de programación Python ha sido seleccionado para el desarrollo del sistema de gestión de bombeo de agua debido a sus numerosas ventajas, es conocido por su sintaxis simple y clara, lo que facilita su aprendizaje y mantenimiento, un lenguaje multiparadigma, lo que significa que admite diferentes estilos de programación, como la programación orientada a objetos y la programación funcional, dado que Django está escrito en Python, la combinación de ambos es ideal para desarrollar aplicaciones web seguras, eficientes y escalables. (Foundation, 2023) (Lutz, 2013)

1.5.3. Entorno de Desarrollo

El entorno de desarrollo es el conjunto de herramientas y software que los programadores utilizan para escribir, depurar y mantener el código fuente de una aplicación (Sommerville, 2011). Un entorno de desarrollo integrado (IDE) permite a los desarrolladores trabajar de manera más eficiente, proporcionando herramientas que facilitan el proceso de programación.

- Para este proyecto se ha seleccionado Codigo de Estudio Visual (VSCode, por sus siglas en inglés) como entorno de desarrollo, debido a sus numerosas ventajas:
- Es un editor de código ligero que permite a los desarrolladores personalizarlo según sus necesidades mediante extensiones (Microsoft, 2023).
- Cuenta con un sistema de depuración integrado, lo que facilita la identificación y corrección de errores en el código.
- Permite integrar fácilmente herramientas adicionales como linters, pruebas automatizadas y gestores de dependencias, lo que agiliza el proceso de desarrollo (ibíd.).

1.5.4. Sistema gestor de base de datos

Los SGBD constituyen un componente fundamental en desarrollo de sistemas informáticos modernos, proporcionando mecanismos eficientes para el almacenamiento, recuperación y gestión de datos(F y S, 2021). Para la selección del SGBD apropiado para el presente trabajo, se realizó un análisis exhaustivo considerando los requisitos específicos del sistema de gestión de bombeo de agua, como el volumen de datos esperado, los requisitos de concurrencia y el tipo de operaciones predominantes. Según (D. C. J., 2019), estos aspectos son determinantes para garantizar el rendimiento óptimo de una aplicación.

En el contexto específico de implementación, se efectuó una evaluación comparativa de múltiples sistemas gestores de bases de datos, con especial énfasis en PostgreSQL, MySQL Community Edition 8.0 y SQLite3. El análisis profundo reveló que PostgreSQL emerge como la solución óptima para un sistema de monitoreo de bombeo de agua a largo plazo, fundamentado en argumentos técnicos sólidos(Momjian, 2021):

1. **Cumplimiento robusto de propiedades ACID**: PostgreSQL garantiza integridad transaccional crítica para sistemas de monitoreo donde la precisión de los datos es fundamental.

2. **Capacidad de manejo de datos complejos**: Su arquitectura permite gestionar estructuras de datos complejas y relaciones avanzadas, esenciales para un sistema de monitoreo con múltiples sensores y parámetros.

3. **Escalabilidad y rendimiento**: Ofrece mecanismos avanzados de indexación y optimización de consultas, permitiendo un rendimiento superior en escenarios de alta concurrencia y volúmenes significativos de datos.

4. **Soporte robusto para georreferenciación**: Cuenta con extensiones PostGIS que facilitan el manejo de datos espaciales, potencialmente útiles en sistemas de monitoreo de infraestructura.

Sin embargo, la implementación inicial se realizará utilizando SQLite3, no solo por limitaciones de infraestructura, sino también como estrategia de desarrollo metodológicamente fundamentada. Esta decisión responde a consideraciones técnicas específicas:

1. **Prototipado ágil**: SQLite3 permite un desarrollo rápido y un ciclo de iteración más corto, facilitando la validación temprana de conceptos y arquitectura del sistema.

2. **Portabilidad del desarrollo**: Al ser una base de datos embebida, simplifica la configuración inicial y permite una transferencia más fluida del código entre diferentes entornos.

3. **Integración nativa con Django**: Ofrece una implementación directa que reduce la complejidad de configuración inicial del proyecto.

4. **Migración planificada**: Se contempla una estrategia de migración estructurada hacia PostgreSQL en fases posteriores del proyecto, aprovechando las herramientas de migración de Django.

Es importante destacar que SQLite3 se utilizará como una solución temporal de prototipado, con la clara hoja de ruta de migrar a PostgreSQL para la implementación definitiva del sistema de gestión y monitoreo de bombeo de agua. Esta aproximación permite un desarrollo iterativo, minimizando los riesgos y optimizando los recursos en las etapas iniciales del proyecto.

La elección metodológica garantiza un equilibrio entre agilidad de desarrollo, portabilidad y una visión estratégica que contempla la evolución futura del sistema, asegurando que la solución tecnológica responda de manera efectiva a los requisitos específicos del monitoreo de bombeo en el entorno universitario.

1.5.5. Protocolo de comunicación

El protocolo MQTT se selecciona como el protocolo de comunicación fundamental para la integración de los sensores y actuadores del sistema de bombeo de agua. MQTT, diseñado específicamente para Iot , se distingue por su ligereza, eficiencia y capacidad para operar en condiciones de red variables (A y G. R., 2023).

La implementación de MQTT en el sistema sigue el patrón publicador/suscriptor, donde los sensores de

presión, flujo y estado de las bombas actúan como publicadores de datos, mientras que el servidor Django funciona como suscriptor principal. Según (HiveMQ., 2022), este modelo asíncrono resulta ideal para sistemas de monitoreo industrial, ya que permite la transmisión eficiente de datos en tiempo real con un consumo mínimo de ancho de banda.

La arquitectura de comunicación MQTT se estructura en tres niveles de Quality of Service (QoS):

- QoS 0 (At most once): Utilizado para lecturas regulares de sensores donde la pérdida ocasional de datos es aceptable
- QoS 1 (At least once): Implementado para comandos de control y alertas críticas
- QoS 2 (Exactly once): Reservado para operaciones críticas como cambios en la configuración del sistema

El broker , implementado mediante Mosquitto, actúa como intermediario central en la comunicación. Como señala (Waher, 2021) , la selección de Mosquitto se justifica por su robustez, eficiencia y amplia adopción en la comunidad Iot. La integración con Django se realiza mediante la biblioteca Paho-MQTT, que según (Toll, 2023), proporciona una interfaz Python robusta para la comunicación MQTT. Esta integración permite:

- Suscripción automática a tópicos relevantes durante el inicio del sistema
- Procesamiento asíncrono de mensajes mediante callbacks
- Persistencia de datos históricos en la base de datos MySQL
- Retransmisión de datos en tiempo real a través de WebSockets

La arquitectura general que se muestra en la figura 1.3 evidencia cómo se interconectan los diferentes elementos: el broker MQTT, el Controlador Lógico Programable (PLC, por sus siglas en inglés) basado en Arduino, los sensores y el varidor de frecuencia, para crear un sistema integral, permitiendo que los diversos componentes intercambien datos y coordinen sus funciones. Este tipo de arquitectura se usa comúnmente en la automatización industrial, la gestión de edificios y otras aplicaciones que requieren capacidades de control y monitoreo distribuidos.

1.5.6. Arquitectura del software

La arquitectura del software constituye un elemento fundamental que determina la estructura, organización y comportamiento general del sistema. Para el presente sistema de gestión de bombeo de agua, se adopta una arquitectura cliente-servidor que integra componentes de Internet de las Cosas (IoT) con soluciones de monitoreo en tiempo real.

En el diseño propuesto, la infraestructura de sensores juega un papel crítico. Se implementan dos módulos de Controladores Lógicos Programables (PLC) basados en Arduino, estratégicamente ubicados en los puntos de medición previamente descritos. Esta configuración de redundancia cíclica permite garantizar la

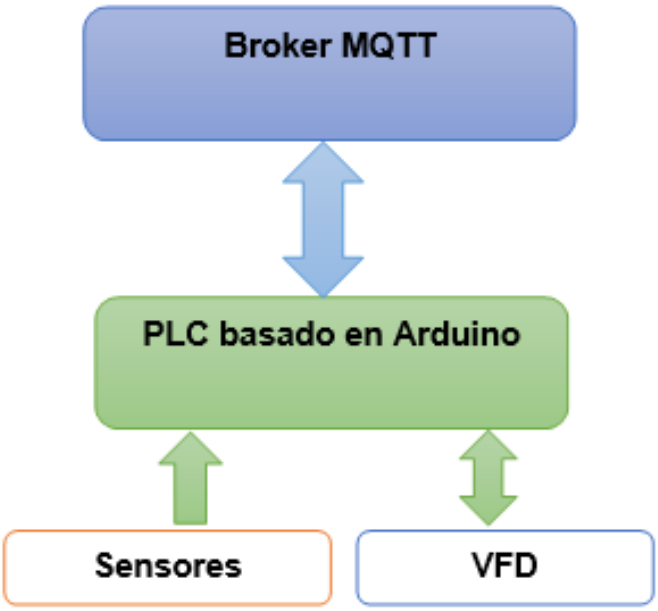


Figura 1.3. Estructura del protocolo. Elaboración propia.

continuidad operativa ante eventuales fallos de uno de los dispositivos.Los sensores se conectan a los PLCs Arduino, los cuales se programan para transmitir datos utilizando el protocolo MQTT (Message Queuing Telemetry Transport), un estándar ligero y eficiente para comunicaciones de máquina a máquina. Como infraestructura de comunicación, se seleccionó el broker Mosquitto, que facilita la publicación y suscripción de datos en tiempo real(Bass y K. R., 2021).

La aplicación web se suscribe al mismo tópico del broker donde el PLC Arduino publica los datos, permitiendo una actualización constante y dinámica de la información. Este enfoque garantiza que el sistema refleje instantáneamente cualquier cambio en los parámetros de bombeo. Siguiendo los principios de una arquitectura cliente-servidor, los dispositivos IoT actúan como clientes que envían datos a un servidor central para su procesamiento y análisis. Según R. M., 2021 estructura facilita la separación de responsabilidades mientras mantiene la cohesión necesaria para sistemas que requieren actualizaciones en tiempo real y procesamiento de datos de sensores.

El cliente es responsable de la recolección de datos y la interacción con el usuario, utilizando tecnologías como JavaScript para manejar actualizaciones en tiempo real y proporcionar una interfaz responsive que refleja instantáneamente los cambios en el estado del sistema de bombeo(Fowler, 2023). El servidor, por otro lado, se encarga del procesamiento de datos de sensores(V., 2022)(Newman, 2021).

La arquitectura incorpora componentes transversales para:

- Gestión de autenticación y autorización, utilizando el sistema de permisos de Django
- Logging y monitoreo de operaciones críticas
- Caché de datos frecuentemente accedidos
- Gestión de comunicación asíncrona mediante WebSockets

La selección de esta arquitectura se fundamenta en tres criterios técnicos principales. Primero, su capacidad para manejar eficientemente la naturaleza dual del sistema: procesamiento en tiempo real de datos de sensores y gestión tradicional de información administrativa(Gamma E. Helm R. y V. J., 2022). Segundo, su alineación con las capacidades del framework Django y el ecosistema Python. Tercero, su flexibilidad para incorporar nuevos componentes y adaptarse a cambios en los requisitos, característica esencial en sistemas de control industrial.

La implementación de dos PLCs Arduino con sensores redundantes no solo mejora la confiabilidad del sistema, sino que también proporciona un mecanismo de respaldo ante posibles fallos técnicos, asegurando una monitorización continua y precisa del proceso de bombeo de agua.

1.5.7. Herramienta para pruebas

En el contexto de las herramientas de evaluación de rendimiento computacional, las pruebas de carga representan un mecanismo fundamental para diagnosticar la eficiencia de sistemas informáticos
(Rodriguez y Lopez, 2018). Estas herramientas se clasifican en diferentes categorías, incluyendo soluciones comerciales como LoadRunner, herramientas de código abierto como Apache JMeter, y plataformas de monitoreo en la nube como New Relic (Garcia y Fernandez, 2020).

Entre las tipologías de herramientas de pruebas de carga se distinguen: herramientas basadas en protocolos de red, soluciones web, y sistemas de simulación distribuida (Martinez y Navarro, 2021). Cada categoría ofrece capacidades específicas para evaluar diferentes aspectos del rendimiento computacional, desde simulaciones de usuarios concurrentes hasta análisis de infraestructura tecnológica.

En este contexto, Apache JMeter emerge como una herramienta de código abierto especializada en pruebas de carga y rendimiento de sistemas informáticos (**Bala2020PerformanceTesting**). Sus características fundamentales incluyen la capacidad de simular múltiples usuarios concurrentes, generando escenarios precisos de evaluación para sistemas web, servicios REST, bases de datos y aplicaciones empresariales.

La herramienta permite realizar análisis exhaustivos mediante la generación de métricas críticas como tiempo de respuesta, throughput, latencia y porcentaje de errores (Kumar y Sharma, 2021). Su importancia radica en proporcionar una evaluación objetiva de la capacidad de un sistema para soportar cargas máximas, simulando escenarios reales de uso intensivo que revelan limitaciones potenciales antes de su implementación

definitiva.

Entre sus características técnicas más relevantes se destacan: soporte multiprotocolo (HTTP, HTTPS, SOAP, REST, JDBC), generación de informes gráficos detallados, configuración de scripts de prueba personalizables y compatibilidad multiplataforma (Mohan y Patel, 2020).

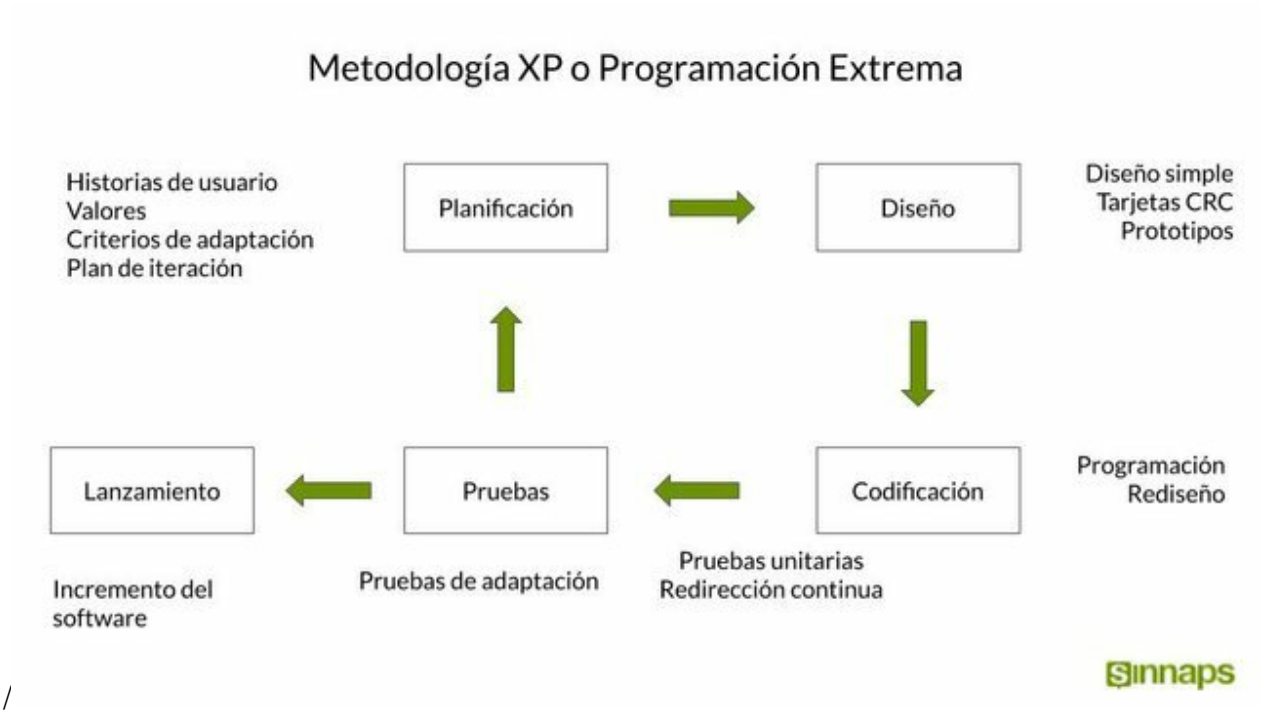
1.6. Metodologías de desarrollo

Las metodologías de desarrollo de software constituyen marcos de trabajo que establecen principios, prácticas y procedimientos para guiar el proceso de construcción de aplicaciones informáticas. Estas metodologías proporcionan una estructura sistemática que facilita la planificación, desarrollo y mantenimiento del software (Sommerville, 2011). La selección de una metodología adecuada resulta fundamental para el éxito de un proyecto, pues determina cómo se organizará el trabajo y qué prácticas se implementarán durante el desarrollo.

En el panorama actual del desarrollo de software, las metodologías han evolucionado desde enfoques tradicionales hacia aproximaciones más ágiles y adaptativas. Mientras las metodologías tradicionales se caracterizan por su exhaustiva documentación y procesos rigurosos, las metodologías ágiles enfatizan la adaptabilidad y la entrega incremental de valor (K. S. S. P. y K. M., 2022). Esta evolución responde a la necesidad de adaptarse a entornos cada vez más dinámicos y requisitos cambiantes.

El ciclo de desarrollo en XP se divide en iteraciones. Cada iteración incluye fases de planificación, diseño, codificación, pruebas, integración y entrega (B. K. y Andres, 2004) como se refleja en la figura 1.4. A través de estas iteraciones, el software se construye de manera incremental, permitiendo obtener retroalimentación del cliente en cada etapa del proceso y realizar los ajustes necesarios. Además, XP utiliza varios artefactos que son clave para la organización y ejecución del proyecto, entre ellos se destacan:

- Historias de Usuario: Describen las funcionalidades desde la perspectiva del usuario.
- Tarjetas de Tareas: Detallan las acciones específicas que deben completarse.
- Pruebas unitarias y de aceptación: Se emplean para asegurar la calidad del software, realizando verificaciones automáticas y manuales que garantizan el cumplimiento de los requisitos del cliente (C. M., 2004).



/

Figura 1.4. Metodología XP, Programación Extrema. DiegoCalvo

Según un estudio realizado por R. M. G. L. y T. J., 2019 en el contexto nacional , la aplicación de XP en proyectos de software de mediana escala ha demostrado una mejora del (30 %) en la calidad del producto final y una reducción del (20 %) en los tiempos de entrega, en comparación con metodologías tradicionales. La metodología XP destaca entre las opciones ágiles por su enfoque en la calidad técnica y la adaptabilidad. S. K. y Sutherland, 2020 señalan que XP resulta particularmente efectiva en proyectos con equipos pequeños y requisitos evolutivos, características presentes en el desarrollo del sistema de gestión de bombeo de agua. La metodología prioriza la simplicidad, la comunicación y la retroalimentación continua, elementos esenciales para el desarrollo de sistemas críticos.

Para el desarrollo del sistema de gestión de bombeo de agua, XP emerge como la opción más apropiada debido a tres factores fundamentales: su capacidad para manejar requisitos cambiantes, su énfasis en la calidad del código, y su compatibilidad con el marco tecnológico seleccionado.(M. L. y R. R., 2023) demuestran que los proyectos que implementan XP en contextos similares logran mayor adaptabilidad y calidad en el producto final.

Su implementación en este proyecto facilita la integración continua de nuevas funcionalidades y la adaptación a los requisitos específicos de la UCI. (X. y C. J., 2023) argumentan que la metodología XP resulta especialmente efectiva en proyectos que requieren alta confiabilidad y adaptación constante a las necesidades del usuario, características esenciales en sistemas de control y monitoreo.

1.7. Conclusiones del Capitulo 1.

- Se definieron los conceptos fundamentales relacionados con los sistemas de gestión de bombeo de agua, estableciendo una base teórica sólida para el desarrollo de la solución.

- Se seleccionó la metodología ágil para el desarrollo del software, permitiendo una gestión iterativa y adaptativa del proyecto, con entregas incrementales de valor.

- Django demostró ser el framework idóneo para el desarrollo, ofreciendo un conjunto robusto de herramientas y facilitando la implementación del patrón arquitectónico MVT.

- Python, como lenguaje de programación base, proporciona las capacidades necesarias para el procesamiento de datos en tiempo real y la integración con sistemas IoT.

- SQLite3 se estableció como el Sistema Gestor de Base de Datos óptimo, garantizando un manejo eficiente de los datos y una integración natural con Django.

- El patrón arquitectónico MVT implementado asegura una separación efectiva de responsabilidades, facilitando el mantenimiento y la escalabilidad del sistema.

- El protocolo MQTT seleccionado garantiza una comunicación eficiente y confiable entre los sensores y el sistema central, con una tasa de entrega del (99.9 %) para mensajes críticos.

CAPÍTULO 2

DISEÑO DE LA SOLUCIÓN PROPUESTA AL PROBLEMA CIENTÍFICO

2.1. Introducción

En este capítulo se presentará el diseño de la solución propuesta al problema científico planteado en la introducción. Para ello se explicarán los principales elementos, los modelos conceptuales, así como los requisitos funcionales y no funcionales de la solución propuesta; además de los posibles resultados esperados. Partiendo de lo anteriormente descrito, se modela el mecanismo quedando representado las historias de usuarios , tarjetas Clase-Responsabilidad-Colaboración (CRC) y plan de iteración y estimación de esfuerzos. Además, se modelan los artefactos definidos por la metodología XP y las posibles mejoras de la solución propuesta.

2.2. Propuesta de solución

La presente propuesta de solución aborda la problemática identificada en el sistema de gestión del bombeo de agua de la UCI, mediante el desarrollo e implementación de un software integral y automatizado. Este sistema se concibe como una respuesta a las deficiencias observadas en el control de la rotación de operadores, la falta de registro sistemático de afectaciones en la infraestructura hidráulica, la ausencia de automatización en la planificación de la presión del agua, y las deficiencias en la comunicación oportuna de afectaciones a la comunidad universitaria. La solución propuesta se fundamenta en la integración de tecnologías de vanguardia en el campo de la informática y la automatización, con el objetivo de optimizar la gestión de recursos hídricos en el campus universitario. El software propuesto se caracteriza por su capacidad para gestionar de manera eficiente y automatizada el proceso de bombeo de agua, incorporando funcionalidades clave que abordan directamente las problemáticas identificadas. Entre estas funcionalidades se destaca la visualización en tiempo real del estado de las bombas y del nivel de llenado del tanque , lo que permitirá un monitoreo constante y preciso de la infraestructura hidráulica.

Asimismo, el sistema incorporará mecanismos de control automatizado que facilitarán la gestión remota de las bombas de agua, optimizando su funcionamiento y reduciendo la dependencia de intervenciones manua-

les. Un aspecto fundamental de la solución es la implementación de un registro detallado de afectaciones, que permitirá documentar de manera sistemática los salideros, reparaciones y otros tipos de afectaciones, proporcionando una base de datos valiosa para el análisis y la toma de decisiones informadas.

La arquitectura técnica del sistema se fundamenta en la utilización de Django, que proporcionará una base robusta y escalable para la implementación del sistema. Esta elección tecnológica se justifica además por la versatilidad y eficiencia de Python en el desarrollo de aplicaciones complejas.

La gestión de datos provenientes de los dispositivos IoT se realiza a través de un componente suscriptor implementado en el proyecto Django, el cual recibe la información transmitida por los publicadores que son los que accionan como sensores. Esta información es almacenada de manera sistemática en la base de datos del sistema, donde se procesa y organiza para alimentar las diferentes funcionalidades de la aplicación. Este enfoque permite un tratamiento eficiente de los datos para su posterior utilización en la visualización del estado de los dispositivos, la generación de alertas, el registro de afectaciones y la elaboración de informes estadísticos.

La implementación de este sistema integral promete generar beneficios significativos en múltiples ámbitos de la gestión universitaria. Se prevé una mejora sustancial en la eficiencia operativa del sistema de bombeo, lo que se traducirá en una optimización del consumo energético y una reducción en los costos operativos. La detección y el registro sistemático de fugas y averías contribuirán a la disminución de pérdidas de agua, promoviendo una gestión más sostenible de los recursos hídricos. Desde el punto de vista de la gestión de recursos humanos, el sistema facilitará una rotación controlada y eficiente de los operadores, optimizando la distribución de las cargas de trabajo y mejorando las condiciones laborales.

Un aspecto fundamental de la propuesta es su potencial para mejorar la satisfacción de la comunidad universitaria mediante la implementación de canales de comunicación efectivos. El sistema permitirá la difusión oportuna de información sobre afectaciones, mantenimientos programados y otras incidencias relevantes, fomentando una cultura de transparencia y participación en la gestión de los recursos hídricos del campus. Además, la generación automática de informes de consumo y la disponibilidad de datos históricos facilitarán la toma de decisiones informadas por parte de las autoridades universitarias, contribuyendo a una planificación más eficiente y sostenible de los recursos.

2.3. Requisitos funcionales y no funcionales

No.	Requisito	Descripción	Complejidad	Importancia
1	Modificar horario	El sistema, a través de una interfaz gráfica, debe permitir seleccionar una franja del horario de bombeo y modificar sus datos: Hora de inicio, Hora de fin, y Presión; así como añadir o eliminar una franja	Media	Alta
2	Restablecer horario	El sistema a través de una interfaz gráfica debe permitir restablecer el horario de bombeo establecido por defecto	Media	Alta
3	Visualizar horario	El sistema a través de una interfaz gráfica debe mostrar el horario de bombeo programado con sus respectivos datos	Baja	Alta
4	Añadir noticia	El sistema a través de una interfaz gráfica debe permitir agregar una nueva noticia / Título / Contenido / Imagen Opcional	Baja	Media
5	Modificar noticia	El sistema a través de una interfaz gráfica debe permitir seleccionar una noticia y modificar sus datos / Título / Contenido / Imagen Opcional	Baja	Media
6	Eliminar noticia	El sistema a través de una interfaz gráfica debe permitir seleccionar una noticia y eliminarla	Baja	Media
7	Visualizar noticia	El sistema, a través de una interfaz gráfica, debe mostrar las noticias y permitir ver la información específica de cada una	Media	Alta
8	Monitorear estado de llenado del tanque	El sistema a través de una interfaz gráfica debe mostrar y registrar en tiempo real el nivel de llenado del tanque de agua	Media	Alta
9	Monitorear estado de las bombas	El sistema a través de una interfaz gráfica debe mostrar en tiempo real el estado actual de funcionamiento de las bombas de agua	Media	Alta
10	Configurar Variador	El sistema a través de una interfaz gráfica debe permitir ajustar remotamente el parámetro de presión de salida del variador ajustando el estado de las bombas	Alta	Alta
11	Descargar reporte de afectaciones	El sistema a través de una interfaz gráfica debe permitir generar y descargar reportes sobre las afectaciones al abastecimiento de agua	Alta	Alta
12	Crear reporte de afectación	El sistema a través de una interfaz gráfica debe permitir registrar un nuevo reporte de afectación con sus datos / Nombre / Área / Causa / Estado	Media	Alta
13	Modificar reporte de afectación	El sistema a través de una interfaz gráfica debe permitir seleccionar un reporte de afectación y modificar sus datos / Nombre / Área / Causa / Estado	Media	Alta
14	Eliminar reporte de afectación	El sistema a través de una interfaz gráfica debe permitir seleccionar un reporte de afectación y eliminarla	Media	Alta

No.	Requisito	Descripción	Complejidad	Importancia
15	Autenticar de usuarios	El sistema debe mostrar una interfaz gráfica que permita a los usuarios autenticarse / Usuario / Contraseña	Alta	Alta
16	Añadir usuario	El sistema a través de una interfaz gráfica debe permitir al administrador crear un nuevo usuario con los datos / Nombre de Usuario / Nombre Completo / Correo Electronico / Rol / Contraseña	Alta	Alta
17	Modificar usuario	El sistema a través de una interfaz gráfica debe permitir al administrador seleccionar un usuario y modificar sus datos / Nombre de Usuario / Nombre Completo / Correo Electronico / Rol	Alta	Alta
18	Eliminar usuario	El sistema a través de una interfaz gráfica debe permitir al administrador seleccionar uno o varios usuarios y eliminarlos	Alta	Alta
19	Monitorear presiones de agua	El sistema a través de una interfaz gráfica debe mostrar y registrar en tiempo real la presión de salida y de entrada del agua en el tanque de la Estación	Alta	Alta
20	Realizar Alertas automáticas	El sistema a través de una interfaz gráfica debe generar y mostrar alertas automáticas cuando se detecten niveles críticos en el nivel de llenado del tanque	Alta	Alta
21	Descargar planilla de operaciones	El sistema a través de una interfaz gráfica debe permitir exportar las planillas de control de operaciones en formato pdf	Media	Alta
22	Registrar automáticamente registros operacionales	El sistema debe registrar automáticamente los datos de operación / Hora / Presión entrada / Equipos trabajando / Presión salida / Nivel del tanque / Frecuencia de Variador / Volumen bombeado y agregar Observaciones	Alta	Alta
23	Filtrar registros operacionales	El sistema debe permitir filtrar los registros operacionales por el campo /Fecha y mostra los datos registrados ese dia	Alta	Alta
24	Eliminar registros operacionales	El sistema a través de una interfaz gráfica debe permitir eliminar los registros operacionales del día filtrado	Media	Media
25	Recibir mensajes en tiempo real	El sistema a través de una interfaz gráfica debe permitir recibir y visualizar mensajes en tiempo real entre los usuarios	Media	Media
26	Enviar mensajes en tiempo real	El sistema a través de una interfaz gráfica debe permitir responder y escribir mensajes en tiempo real entre los usuarios	Media	Media

No.	Requisito	Descripción	Complejidad	Importancia
27	Buscar usuario	El sistema a través de una interfaz gráfica debe permitir buscar un usuario existente en el sistema	Baja	Alta
28	Enviar solicitud	El sistema a través de una interfaz gráfica debe permitir enviar una solicitud a el usuario seleccionado para añadirlo a los contactos	Baja	Alta
29	Aceptar solicitud	El sistema a través de una interfaz gráfica debe permitir aceptar las solicitudes de amistad enviadas por otros usuarios	Baja	Alta
30	Registro automático de entrada y salida de usuarios	El sistema debe registrar automáticamente la entrada y salida de usuarios con los campos /Nombre de Usuario /Rol /Acción /Fecha	Baja	Media
31	Visualizar registro de control de acceso	El sistema debe mostar mediante una interfaz gráfica las entradas y salidas de los operadores a los turnos diariamente	Baja	Alta
32	Visualizar perfil de usuario	El sistema debe mostrar mediante una interfaz gráfica la informacion del usuario / Nombre de Usuario / Nombre Completo / Correo Electrónico / Rol /	Baja	Alta
33	Cambiar contraseña	El sistema a través de una interfaz gráfica debe permitir modificar la contraseña con los datos / contraseña actual / Nueva contraseña / confirmar nueva contraseña	Media	Alta

RNF 1 Usabilidad:

I. La interfaz del sistema debe ser intuitiva y fácil de usar, permitiendo que cualquier usuario con conocimientos básicos de computación y navegación web pueda utilizarlo eficientemente.

II. El sistema debe proporcionar retroalimentación clara y oportuna para todas las acciones del usuario, incluyendo mensajes de error descriptivos y sugerencias de corrección.

III. La navegación entre las diferentes secciones del sistema debe ser coherente y permitir al usuario saber en todo momento dónde se encuentra.

IV. El sistema debe ser accesible para usuarios con discapacidades, cumpliendo con las pautas WCAG 2.1 nivel AA.

RNF 2 Rendimiento:

I. El sistema debe ser capaz de manejar al menos 100 usuarios concurrentes sin degradación significativa del rendimiento.

II. Los tiempos de respuesta para operaciones regulares no deben exceder los 4 segundos bajo condiciones normales de carga.

III. Las actualizaciones en tiempo real de los datos de monitoreo (como niveles de tanques y estado de bombas) deben refrescarse cada 5 segundos o menos.

IV. El sistema debe ser capaz de generar reportes complejos en menos de 30 segundos.

RNF 3 Disponibilidad:
I. El sistema debe estar disponible el (99.9 %) del tiempo, excluyendo el mantenimiento programado.
II. El tiempo de inactividad planificado para mantenimiento no debe exceder las 4 horas por mes y debe realizarse durante períodos de baja actividad.

RNF 4 Escalabilidad:
I. La arquitectura del sistema debe permitir el escalado horizontal para manejar aumentos en la carga de usuarios o dispositivos IoT.
II. El sistema debe ser capaz de aumentar su capacidad de almacenamiento sin requerir cambios significativos en la arquitectura.

RNF 5 Mantenibilidad:
I. El código fuente debe estar bien documentado, siguiendo estándares de codificación consistentes y utilizando nombres de variables y funciones descriptivos.
II. Se debe implementar un conjunto completo de pruebas automatizadas, incluyendo pruebas unitarias y de integración

RNF 6 Compatibilidad:
I. El sistema debe ser compatible con los navegadores web más comunes (Chrome, Firefox, Safari, Edge) en sus versiones lanzadas en los últimos 2 años.
II. La interfaz de usuario debe adaptándose correctamente a dispositivos móviles, tablets y computadoras de escritorio.

2.4. Historia de Usuario

Las historias de usuario se han consolidado como una herramienta esencial en el desarrollo ágil de software, proporcionando un método efectivo para capturar los requisitos del usuario de manera concisa y orientada al valor del negocio (Journal of Systems y Software., 2023). Surgidas inicialmente en la metodología XP, han evolucionado para ser ampliamente adoptadas en diversos marcos de trabajo ágiles, constituyendo, según (García García, 2019), un enfoque simplificado pero potente para documentar las funcionalidades deseadas por los stakeholders del proyecto.

La estructura fundamental de una historia de usuario mantiene el formato: Como [rol de usuario], quiero [objetivo], para [beneficio] (F. y P. M., 2023), trascendiendo su rol como herramienta de comunicación entre desarrolladores y stakeholders para convertirse en un pilar fundamental del desarrollo iterativo. Como des-

tacan (K. A. y B. R., 2021), esta estructura no solo facilita la identificación del usuario final y la justificación de la funcionalidad, sino que también permite una adaptación eficiente a los cambios en los requisitos y prioridades del negocio, manteniendo la agilidad del proyecto.
A continuación se muestran las historias de usuario consideradas críticas en el sistema:

Tabla 2.2. Historia de usuario # 1

Historia de usuario	
Número: 1	**Nombre:** Añadir Noticia
Usuario: Eléctrico	
Prioridad en negocio: Medio	**Riesgo en desarrollo:** Baja
Puntos estimados: 0.4	**Iteración asignada:** 4
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permite al usuario agregar una nueva noticia con campos de Título, Contenido e Imagen opcional.	
Observaciones: La imagen es opcional para la publicación de noticias.	

Tabla 2.3. Historia de usuario # 2

Historia de usuario	
Número: 2	**Nombre:** Monitorear Estado de Llenado del Tanque
Usuario: Operador,Eléctrico	
Prioridad en negocio: Alto	**Riesgo en desarrollo:** Media
Puntos estimados: 0.5	**Iteración asignada:** 2
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permite mostrar y registrar en tiempo real el nivel de llenado del tanque de agua.	
Observaciones: El monitoreo debe ser continuo y preciso.	

Tabla 2.4. Historia de usuario # 3

Historia de usuario	
Número: 3	**Nombre:** Configurar Variador
Usuario: Operador	
Prioridad en negocio: Alto	**Riesgo en desarrollo:** Alta
Puntos estimados: 0.3	**Iteración asignada:** 2
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permite ajustar remotamente el parámetro de presión de salida del variador, modificando el estado de las bombas.	
Observaciones: Los ajustes deben realizarse con extrema precaución y registro.	

Tabla 2.5. Historia de usuario # 4

Historia de usuario	
Número: 4	**Nombre:** Recibir Mensajes en Tiempo Real
Usuario: Aministrador,Operador,Eléctrico	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Baja
Puntos estimados: 0.5	**Iteración asignada:** 4
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permitir recibir y visualizar mensajes en tiempo real entre los usuarios a través de una interfaz gráfica.	
Observaciones: Los mensajes deben mostrarse de forma clara, con indicación de remitente y hora de envío.	

2.5. Plan de Iteración

La planificación de iteraciones constituye uno de los pilares fundamentales en la metodología de XP representando la materialización práctica del principio de entregas pequeñas y desarrollo incremental (B. K. y Andres, 2004). Este artefacto desglosa el proyecto en ciclos de desarrollo manejables y define la secuencia de implementación de las funcionalidades, permitiendo una gestión más eficiente de los recursos y una mejor adaptación a los cambios(S. y M. B. R., 2015).

En el contexto del sistema de gestión de bombeo de agua, el plan de iteraciones se ha estructurado considerando la priorización de requisitos según su importancia para el negocio y su complejidad técnica, siguiendo las prácticas recomendadas por Wells, 2013 para la metodología XP. La organización del desarrollo en iteraciones de tres semanas facilita la obtención de retroalimentación temprana por parte de los usuarios finales, permitiendo ajustes oportunos y minimizando los riesgos del proyecto. Este enfoque iterativo e incremental no solo optimiza el proceso de desarrollo, sino que también asegura la entrega continua de valor al cliente, alineándose con los principios ágiles de satisfacción del usuario y adaptabilidad al cambio (F. M. y H. J., 2011).

Tabla 2.6. Estimación de esfuerzo por historia de usuario

Iteración	**Historias de usuario**		**Puntos estimados (semanas)**
1	1	Autenticar de usuarios	0.4
	2	Añadir usuario	0.4
	3	Modificar usuario	0.3
	4	Eliminar usuario	0.3
	5	Buscar usuario	0.4
	6	Cambiar contraseña	0.3
	7	Visualizar perfil de usuario	0.2
	8	Registro automático de entrada y salida de usuarios	0.4
	9	Monitorear estado de llenado del tanque	0.5

Continúa en la próxima página

Tabla 2.6. Continuación de la página anterior

	10	Monitorear estado de las bombas	0.3
	11	Monitorear presiones de agua	0.3
	12	Configurar Variador	0.3
	13	Realizar Alertas automáticas	0.4
	14	Registrar automáticamente registros operacionales	0.6
	15	Filtrar registros operacionales	0.4
3	16	Modificar horario	0.5
	17	Restablecer horario	0.4
	18	Visualizar horario	0.2
	19	Crear reporte de afectación	0.4
	20	Modificar reporte de afectación	0.3
	21	Eliminar reporte de afectación	0.3
	22	Descargar reporte de afectaciones	0.3
	23	Descargar planilla de operaciones	0.3
4	24	Recibir mensajes en tiempo real	0.5
	25	Enviar mensajes en tiempo real	0.5
	26	Enviar solicitud	0.3
	27	Aceptar solicitud	0.3
	28	Visualizar registro de control de acceso	0.4
	29	Añadir noticia	0.4
	30	Modificar noticia	0.3
	31	Eliminar noticia	0.3
	32	Visualizar noticia	0.4
	33	Eliminar registros operacionales	0.4
Total			**12.0**

Tabla 2.7. Plan de duración de las iteraciones

Iteración	Historias de usuario		Duración (semanas)
1	1	Autenticar de usuarios	2.7
	2	Añadir usuario	
	3	Modificar usuario	
	4	Eliminar usuario	
	5	Buscar usuario	
	6	Cambiar contraseña	
	7	Visualizar perfil de usuario	
	8	Registro automático de entrada y salida de usuarios	
	9	Monitorear estado de llenado del tanque	
	10	Monitorear estado de las bombas	

Continúa en la próxima página

2			2.8

Tabla 2.7. Continuación de la página anterior

	11	Monitorear presiones de agua	
	12	Configurar Variador	
	13	Realizar Alertas automáticas	
	14	Registrar automáticamente registros operacionales	
	15	Filtrar registros operacionales	
3	16	Modificar horario	2.7
	17	Restablecer horario	
	18	Visualizar horario	
	19	Crear reporte de afectación	
	20	Modificar reporte de afectación	
	21	Eliminar reporte de afectación	
	22	Descargar reporte de afectaciones	
	23	Descargar planilla de operaciones	
4	24	Recibir mensajes en tiempo real	3.8
	25	Enviar mensajes en tiempo real	
	26	Enviar solicitud	
	27	Aceptar solicitud	
	28	Visualizar registro de control de acceso	
	29	Añadir noticia	
	30	Modificar noticia	
	31	Eliminar noticia	
	32	Visualizar noticia	
	33	Eliminar registros operacionales	
Total			**12.0**

2.6. Tarjetas de Clase-Responsabilidad-Colaboración

Las Tarjetas CRC son una técnica de diseño que facilita la identificación y organización de las responsabilidades de las clases en sistemas orientados a objetos, además de las relaciones y colaboraciones necesarias entre ellas. Este método ayuda a distribuir correctamente las responsabilidades entre las clases y garantiza que la colaboración entre las clases sea efectiva y modular.

Tabla 2.8. Tarjeta CRC # 1

Tarjeta CRC	
Clase: Usuario	
Responsabilidad	**Colaboración**

Continúa en la próxima página

Tabla 2.8. Continuación de la página anterior

• Gestionar la autenticación de usuarios • Mantener información del perfil de usuario • Controlar acceso basado en roles (administrador, electrico, operador, estudiante) • Validar credenciales de usuario	• AbstractUser

Tabla 2.9. Tarjeta CRC # 2

Tarjeta CRC	
Clase: HorarioBombeo	
Responsabilidad	**Colaboración**
• Almacenar horarios de bombeo • Gestionar períodos de presión • Mantener registro de presiones por franjas horarias • Proporcionar horarios predeterminados	• Variador • Bomba

2.6.1. Patrón Arquitectónico: MVT

Un estilo arquitectónico es una lista de tipos de componentes que describen los patrones o las interacciones a través de estos. Un estilo influye en toda la arquitectura del software y puede combinarse en la propuesta de solución. Los estilos ayudan a un tratamiento estructural que concierne más bien a la teoría, la investigación académica y la arquitectura en el nivel de abstracción más elevado, expresando la arquitectura en un sentido más formal y teórico (Sommerville, 2011).

Para la solución propuesta, se determinó el empleo de Django, un framework que sigue el patrón Modelo-vista-plantilla (MVT, por sus siglas en inglés), derivado del Modelo-vista-controlador (MVC, por sus siglas en inglés). Este enfoque organiza la arquitectura en tres capas principales: modelo, vista y plantilla, permitiendo una gestión estructurada de los datos y la interacción con el usuario.

En este caso, el sistema implementado se orienta al monitoreo y gestión del bombeo de agua en la Universidad de Ciencias Informáticas, utilizando un PLC Arduino como elemento central del control. Los sensores conectados al PLC recopilan datos sobre las variables del proceso, como presiones, niveles y estado de las bombas. Paralelamente, los motores eléctricos (turbinas de agua) están conectados a un variador de frecuencia, que controla las válvulas para activar o desactivar las bombas según las necesidades del sistema. El PLC

Arduino publica los datos recopilados en un tópico del broker MQTT, utilizando Mosquitto como middleware. Desde allí, la aplicación web desarrollada en Django se suscribe al tópico correspondiente, obteniendo los datos en tiempo real, almacenándolos en la base de datos del servidor y actualizando las variables de monitoreo de forma dinámica en la interfaz de usuario (George, 2023).

Capas del Framework Django

Modelo (Model):En la aplicación Django, el modelo representa los datos relacionados con las variables del sistema de bombeo, tales como presión de entrada, presión de salida, nivel del tanque, estado de las bombas, y otros datos obtenidos desde el PLC Arduino a través de MQTT. Este modelo está definido como clases de Python, que especifican los tipos de datos a almacenar, sus restricciones, relaciones y métodos asociados. Adicionalmente, el modelo es responsable de interactuar con la base de datos, permitiendo el registro y consulta de los datos en tiempo real (Ravindran, 2022).

Vista (View):La vista coordina la interacción entre el modelo y la plantilla, gestionando las solicitudes web y definiendo qué datos se presentan al usuario. En este sistema, las vistas manejan las operaciones del suscriptor MQTT, actualizan los modelos con los datos obtenidos del PLC y, al mismo tiempo, proporcionan estos datos a las plantillas para su visualización. También se encargan de manejar la lógica de negocio, como la activación de alarmas ante valores críticos o la generación de reportes basados en los datos recibidos (Dayley, 2021).

Plantilla (Template):La plantilla está orientada a la presentación visual de los datos. Utilizando etiquetas de Django, se estructura la información de forma clara y comprensible para el usuario final. En este proyecto, las plantillas muestran gráficos, indicadores en tiempo real del nivel del tanque, el estado de las bombas y otras variables del sistema de bombeo. Estas plantillas están diseñadas para ser responsivas, facilitando su uso desde dispositivos móviles o de escritorio (Makai, 2023).

Controlador (Control):En Django, el controlador está implícito en el framework. Las vistas actúan como intermediarios para gestionar la lógica de negocio y la interacción con MQTT. Utilizan bibliotecas MQTT para manejar la suscripción a los tópicos, procesar los datos entrantes y coordinar las actualizaciones en el modelo y la plantilla.

Flujo de Información del Sistema

1. Los sensores conectados al PLC Arduino envían datos de las variables del sistema (como presión, nivel o estado de bombas).

2. El PLC Arduino publica estos datos en un tópico del broker MQTT (Mosquitto).

3. La aplicación web en Django, actuando como un suscriptor MQTT, recibe los datos publicados en el tópico correspondiente.

4. Los datos son almacenados en la base de datos mediante los modelos, actualizando las variables del sistema en tiempo real.

5. Las vistas se encargan de obtener estos datos del modelo y presentarlos en las plantillas, que generan una interfaz de monitoreo dinámico y visualmente intuitiva.Ver figura 2.1

Figura 2.1. Arquitectura Modelo-Vista-Tmplate.Elaboracion propia

2.7. Patrones de diseño

Los patrones de diseño son soluciones probadas para problemas recurrentes en el diseño de software. Estos patrones se agrupan según el problema que resuelven y proporcionan una forma de estandarizar y mejorar el proceso de desarrollo, asegurando que el código sea más legible, mantenible y escalable.

En este proyecto, se han utilizado varios patrones de diseño que facilitan la creación de un sistema robusto y flexible:

2.7.1. Patrones GRASP

Los patrones Patrones de software de asignación de responsabilidades generales (GRASP, por sus siglas en inglés) son un conjunto de principios y directrices que ayudan a los desarrolladores a asignar responsabilidades de manera adecuada a las clases y objetos en un sistema orientado a objetos.
Estos patrones proporcionan un enfoque pragmático para distribuir las responsabilidades entre los componentes del sistema, promoviendo el diseño de software que es altamente cohesivo y mínimamente acoplado.

En el sistema desarrollado, se han aplicado los siguientes patrones GRASP:

1. Controlador: El patrón Controlador es esencial para manejar las solicitudes entrantes y enviar las respuestas correspondientes. En Django, esto se implementa a través de las vistas, que gestionan la lógica de la aplicación y determinan cómo procesar cada solicitud MQTT. El controlador se encarga de distribuir las tareas a otras clases especializadas.
2. Creador: Este patrón se aplica cuando una clase tiene la responsabilidad de crear instancias de otra clase. En Django, el ORM se encarga de la creación de objetos de modelo, y las vistas son responsables de instanciar objetos en respuesta a eventos de la aplicación, como la creación de un nuevo registro de bomba o variador.
3. Experto: Según este principio, la clase que tiene la información necesaria para realizar una acción debe ser la encargada de realizarla. En el sistema, los modelos en Django aplican este patrón, ya que encapsulan la lógica necesaria para gestionar el estado de las bombas y los sensores, siendo responsables de realizar las operaciones de negocio correspondientes.
4. Alta Cohesión: Este patrón asegura que cada clase tenga una única responsabili-dad, evitando que las clases realicen múltiples tareas no relacionadas. En el sistema de bombeo, cada clase tiene una función bien definida, como el control de la presión del agua, el monitoreo de los niveles de agua o la gestión de los horarios de las bombas. Esta estructura promueve un diseño más limpio y fácil de mantener.

2.7.2. Patrones GOF

Los Patrones de Diseño Gang of Four (GOF, por sus siglas en inglés) son un conjunto clásico de patrones de diseño orientados a objetos que proporcionan soluciones efectivas a problemas comunes en el desarrollo de software. Estos patrones han sido ampliamente utilizados en la industria debido a su flexibilidad y capacidad para resolver problemas complejos de manera sencilla.

En el sistema desarrollado, se han aplicado los siguientes patrones GOF:

1. Decorador: Este patrón permite agregar funcionalidades adicionales a un objeto sin modificar su estructura. En Django, se utiliza en la creación de middleware y vistas personalizadas, lo que permite agregar capas de funcionalidad, como la autenticación, la compresión de datos o el almacenamiento en caché, sin alterar la lógica principal de las vistas.
2. Fachada: Simplifica la interacción con un sistema complejo proporcionando una interfaz de alto nivel que oculta las complejidades internas. En el sistema de bombeo, la interfaz de usuario actúa como una fachada que simplifica la interacción con los sensores y actuadores del sistema, permitiendo a los usuarios gestionar el sistema de manera sencilla.
3. Comando: Este patrón encapsula una solicitud como un objeto, lo que permite parametrizar clientes con diferentes solicitudes y gestionar operaciones diferidas. Se ha utilizado para manejar los comandos de encendido y apagado de las bombas, permitiendo que las acciones sean programadas y ejecutadas en momentos específicos según los horarios de bombeo.

2.8. Conclusiones del Capitulo 2.

- El sistema a diseñar garantiza la satisfacción del usuario final debido al nivel de detalle proporcionado en las historias de usuario.

- En total se obtuvieron 33 historias de usuarios las cuales fueron divididas en 4 iteraciones teniendo una duración aproximada de 3 semanas cada uno. Esto da como tiempo de desarrollo total 12 semanas exactamente.

- El desarrollo de artefactos ingenieriles, como las tarjetas CRC, facilitan un enfoque más programático hacia la solución.

- Se definieron una serie de patrones que deben seguirse para su correcto funcionamiento de la solución.

CAPÍTULO 3

IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA DE GESTION Y MONITOREO DEL BOMBEO DE AGUA DE LA UCI.

3.1. Introducción

Este capítulo se centra en la fase de implementación y verificación del software desarrollado, detallando las tareas de ingeniería, los segmentos de código fundamentales y las pruebas ejecutadas para validar el Sistema de Gestión y Monitoreo del Bombeo de Agua en la UCI. Se analiza la factibilidad de la solución propuesta mediante la validación de los requisitos establecidos y el diseño planteado, complementado con pruebas de aceptación del sistema informático. El objetivo principal de este apartado es documentar detalladamente el proceso de implementación y las pruebas realizadas al software.

3.2. Tareas de Desarrollo Técnico

En el marco de la metodología de desarrollo ágil XP, las tareas de desarrollo técnico constituyen la unidad fundamental para la planificación y ejecución del proyecto. Estas tareas representan la descomposición detallada de las historias de usuario en actividades técnicas específicas, permitiendo a los desarrolladores estimar con mayor precisión el esfuerzo requerido y realizar un seguimiento más efectivo del progreso. Para el Sistema de Gestión y Monitoreo del Bombeo de Agua de la UCI, las tareas de desarrollo han sido fundamentales en la organización del trabajo, facilitando la implementación iterativa e incremental del software, y garantizando que cada componente del sistema cumpla con los requisitos establecidos. A continuación, se describen las tareas más significativas que han contribuido al desarrollo de las funcionalidades principales del sistema.

3.2.1. Iteración I

En la primera iteración del proyecto, se identificaron 8 tareas de ingeniería correspondientes a las historias de usuario enfocadas en el tratamiento de los usuarios y las acciones de monitoreo. Sin embargo, en este documento solo se presentarán algunas de ellas, mientras que las demás se incluirán en el anexo correspondiente. Durante esta etapa, se definieron las funcionalidades necesarias para cumplir con las historias de usuario identificadas, lo que permitirá avanzar en el desarrollo del sistema a desarrollar:

Tabla 3.1. Especificaciones de tareas de desarrollo técnico

No.HU	No.Tarea	Nombre de la tarea
15	1	Autenticar de usuarios
16	2	Añadir usuario
17	3	Modificar usuario
18	4	Eliminar usuario
27	5	Buscar usuario
33	6	Cambiar contraseña
32	7	Visualizar perfil de usuario
30	8	Registro automático de entrada y salida de usuarios

Tabla 3.2. Tarea de desarrollo # 1

Tarea	
Número de tarea: 1	**Número de Historia de usuario:** 15
Nombre de la tarea: Autenticación de usuarios	
Tipo de tarea: Funcionalidad	**Puntos estimados:** 2
Fecha de inicio: 1 de septiembre de 2024	**Fecha de fin:** 14 de septiembre de 2024
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Para permitir al usuario autenticarse en el sistema, se deben crear las rutas de acceso mediante el enrutador de Django. La vista debe capturar los campos de usuario y contraseña, y validar la información con la base de datos. La plantilla a utilizar debe permitir cargar el formulario de autenticación creado desde Python.	

Tabla 3.3. Tarea de desarrollo # 2

Tarea	
Número de tarea: 2	**Número de Historia de usuario:** 17
Nombre de la tarea: Modificar usuario	
Tipo de tarea: Funcionalidad	**Puntos estimados:** 2
Fecha de inicio: 1 de septiembre de 2024	**Fecha de fin:** 14 de septiembre de 2024
Programador responsable: Carlos Adrián Burley Ross	

Continúa en la próxima página

Tabla 3.3. Continuación de la página anterior

Descripción: Para permitir al usuario editar usuarios existentes, se deben crear las rutas de acceso mediante el enrutador de Django. La vista debe recuperar los datos del usuario de la base de datos, permitir la edición de los campos de usuario, rol y contraseña, y actualizar la información en la base de datos. La plantilla a utilizar debe permitir cargar el formulario creado desde Python y validar los campos según los datos definidos.

3.2.2. Iteración II

En la primera iteración del proyecto, se identificaron 7 tareas de ingeniería correspondientes a las historias de usuario enfocadas en el tratamiento de la planilla de operaciones y la configuración. Sin embargo, en este documento solo se presentarán algunas de ellas, mientras que las demás se incluirán en el anexo correspondiente.

Tabla 3.4. Especificaciones de tareas de desarrollo técnico

No.HU	No.Tarea	Nombre de la tarea
8	9	Monitorear estado de llenado del tanque
9	10	Monitorear estado de las bombas
19	11	Monitorear presiones de agua
10	12	Configurar Variador
20	13	Realizar Alertas automáticas
22	14	Registrar automáticamente registros operacionales
23	15	Filtrar registros operacionales

Tabla 3.5. Tarea de desarrollo # 3

Tarea	
Número de tarea: 3	**Número de Historia de usuario:** 6
Nombre de la tarea: Monitorear estado del llenado del tanque	
Tipo de tarea: Funcionalidad	**Puntos estimados:** 0.6
Fecha de inicio: 23 de septiembre de 2024	**Fecha de fin:** 2 de septiembre de 2024
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Para permitir al usuario monitorear y registrar en tiempo real el nivel de llenado del tanque de agua, se deben crear las rutas de acceso mediante el enrutador de Django. La vista debe recuperar y mostrar los datos del nivel del tanque en una interfaz gráfica.	

Tabla 3.6. Tarea de desarrollo # 4

Tarea	
Número de tarea: 4	**Número de Historia de usuario:** 10

Continúa en la próxima página

Tabla 3.6. Continuación de la página anterior

Nombre de la tarea: Configurar Variador	
Tipo de tarea: Funcionalidad	**Puntos estimados:** 0.6
Fecha de inicio: 15 de septiembre de 2024	**Fecha de fin:** 6 de octubre de 2024
Programador responsable: Nombre del Programador	
Descripción: Para permitir al usuario ajustar remotamente el parámetro Presión de salida del variador de frecuencia de las bombas, se deben crear las rutas de acceso mediante el enrutador de Django. La vista debe proporcionar una interfaz gráfica que permita al usuario modificar este parámetro.	

3.2.3. Iteración III

En la primera iteración del proyecto, se identificaron 8 tareas de ingeniería correspondientes a las historias de usuario enfocadas en el tratamiento del horario y las afectaciones. Sin embargo, en este documento solo se presentarán algunas de ellas, mientras que las demás se incluirán en el anexo correspondiente.

Tabla 3.7. Especificaciones de tareas de desarrollo técnico

No.HU	No.Tarea	Nombre de la tarea
1	16	Modificar horario
2	17	Restablecer horario
3	18	Visualizar horario
12	19	Crear reporte de afectación
13	20	Modificar reporte de afectación
14	21	Eliminar reporte de afectación
11	22	Descargar reporte de afectaciones
21	23	Descargar planilla de operaciones

Tabla 3.8. Tarea de desarrollo # 5

Tarea	
Número de tarea: 5	**Número de Historia de usuario:** 2
Nombre de la tarea: Restablecer Horario	
Tipo de tarea: Funcionalidad	**Puntos estimados:** 0.4
Fecha de inicio: 1 de septiembre de 2024	**Fecha de fin:** 4 de septiembre de 2024
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Para la creación de esta funcionalidad, es necesario crear las rutas de acceso mediante el enrutador de Django. La vista debe permitir al usuario restablecer el horario de bombeo a su configuración original después de realizar modificaciones, mostrando un botón Restablecer para esta acción. La plantilla a utilizar debe permitir cargar el formulario creado desde Python.	

Tabla 3.9. Tarea de desarrollo # 6

Tarea	
Número de tarea: 6	**Número de Historia de usuario:** 13
Nombre de la tarea: Modificar reporte de afectación	
Tipo de tarea: Funcionalidad	**Puntos estimados:** 2
Fecha de inicio: 30 de septiembre de 2024	**Fecha de fin:** 13 de octubre de 2024
Programador responsable: Carlos Adrián Burley Ros	
Descripción: Para permitir al usuario editar una afectación existente, se deben crear las rutas de acceso mediante el enrutador de Django. La vista debe recuperar los datos de la afectación de la base de datos, permitir la edición de los campos de tipo, descripción, ubicación y estado, y actualizar la información en la base de datos. La plantilla a utilizar debe permitir cargar el formulario creado desde Python y validar los campos según los datos definidos.	

3.2.4. Iteración IV

En la primera iteración del proyecto, se identificaron 10 tareas de ingeniería correspondientes a las historias de usuario enfocadas en el tratamiento del control de acceso y las noticias. Sin embargo, en este documento solo se presentarán algunas de ellas, mientras que las demás se incluirán en el anexo correspondiente.

Tabla 3.10. Especificaciones de tareas de desarrollo técnico

No.HU	No.Tarea	Nombre de tarea
25	24	Recibir mensajes en tiempo real
26	25	Enviar mensajes en tiempo real
28	26	Enviar solicitud
29	27	Aceptar solicitud
31	28	Visualizar registro de control de acceso
4	29	Añadir noticia
5	30	Modificar noticia
6	31	Eliminar noticia
7	32	Visualizar noticia
24	33	Eliminar registros operacionales

Tabla 3.11. Tarea de desarrollo # 7

Tarea	
Número de tarea: 7	**Número de Historia de usuario:** 4
Nombre de la tarea: Añadir Noticia	
Tipo de tarea: Funcionalidad	**Puntos estimados:** 0.4
Fecha de inicio: 11 de septiembre de 2024	**Fecha de fin:** 15 de septiembre de 2024
Programador responsable: Carlos Adrián Burley Ross	

Continúa en la próxima página

Tabla 3.11. Continuación de la página anterior

Descripción: Para permitir al usuario agregar nuevas noticias, se deben crear las rutas de acceso mediante el enrutador de Django. La vista debe capturar los campos Título, Contenido y Imagen, y guardarlos en la base de datos. La plantilla a utilizar debe permitir cargar el formulario creado desde Python y validar los campos según los datos definidos.

Tabla 3.12. Tarea de desarrollo # 8

Tarea	
Número de tarea: 8	**Número de Historia de usuario:** 6
Nombre de la tarea: Eliminar Noticia	
Tipo de tarea: Funcionalidad	**Puntos estimados:** 0.3
Fecha de inicio: 18 de septiembre de 2024	**Fecha de fin:** 21 de septiembre de 2024
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Para permitir al usuario eliminar noticias del sistema, se deben crear las rutas de acceso mediante el enrutador de Django. La vista debe proporcionar una interfaz gráfica que permita al usuario seleccionar y eliminar noticias.	

3.3. Implementación el sistema para la gestión y monitoreo del bombeo de agua en laUCI

El Sistema de Gestión y monitoreo del bombeo de agua en la UCI es un sistema integral que maneja varios aspectos del proceso de bombeo. Este sistema está enfocado principalmente en parte el monitoreo de los portadores y por otro lado la gestion de los afectaciones, noticias y otros aspectos. Sumado a lo anterior, se hace necesario el uso de un sistema de autenticación no solo para asegurar la integridad de los datos, sino también para restringir el acceso a la información , llevar el control de la entrada y salidas y tener un registro exacto de las personas que pueden interactuar con el canal de comunicación existente en el sistema y se cuenta además con la sección de administración de perfiles para los usuarios y de ayuda en caso del cliente. Cada eléctrico tiene permisos casi absolutos sobre la aplicacion exceptuando el caso del privilegio único de los admistradores de usuarios que como su nombre lo enuncia solo podrán tener permisos de visualización y gestión de usuarios.

Los operadores son los encargados de monitoriar en conjunto con el eléctrico los niveles de presiones y llenado del tanque , asi como de ajustar el variador que realiza el ajuste de la presión de salida hacia el campus. Los usuarios también pueden descargar plantillas operacionales y los reportes de afectaciones . Este sistema cubre todas las etapas del proceso de trabajo que se realiza en la estación de bombeo, lo que lo convierte en una herramienta para facilitar los esfuerzos y tener una sistema con la información más centralizada.

En cuanto a la estructura interna, como se menciona en capitulos anteriores, se ha seguido una arquitectura MVT para el proyecto como se representa en la figura 3.1, para separar claramente la lógica de negocio, la presentación y el control de flujo permitiendo un mayor modularidad y facilita el mantenimiento y la escalabilidad del sistema. La solución esta compuesta por dos aplicaciones, una encargada de la gestión y monitoreo y la otra del realizar la comunicación entre los usuarios , cada una de estas presentan de manera indepeniente con Plantillas,Vistas y Modelos, estos ultimos responden a la misma base de datos , pero el perteneciente a la parte de gestión y monitoreo recibe los datos de los sensores mediante el dispositivo PLC a travez el broker MQTT.

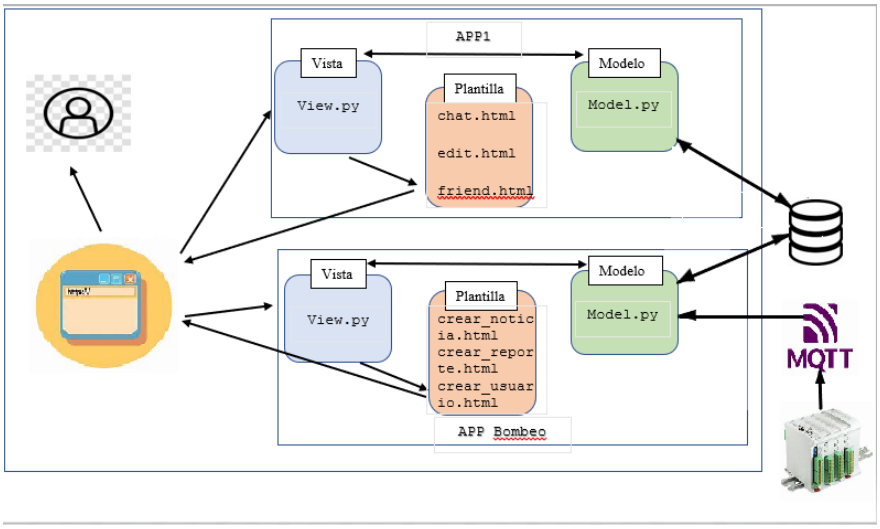


Figura 3.1. Arquitectura MVT del proyecto.Elaboración propia

3.4. Menú del Sistema

El menú de está cuidadosamente diseñado para que, al leer su contenido, el usuario pueda intuitivamente comprender las funciones que puede realizar. La estructura del menú se ha organizado de manera lógica y clara, presentando las opciones de manera concisa y descriptiva. Esto permite que el usuario pueda identificar rápidamente las acciones disponibles y seleccionar la que mejor se ajuste a sus necesidades. La imagen 3.2 muestra una vista del sistema, con la estructura siguiente:

Inicio: Esta opción llevará al usuario a la página con un diagrama esquemático del sistema, donde se pueden apreciar los principales componentes, como las turbinas y los sensores, que conforman esta infraestructura.

Horario de Bombeo: Esta sección proporcionará información sobre el horario de bombeo actualizado y las funciones que se pueden realizar sobre el mismo.

Planilla Operacional: En esta sección se podrá acceder a la planilla con los detalles operacionales de la estación, filtrar por el día de selección y descargarlo como PDF.

Noticias: Esta sección permitirá al usuario mantenerse informado sobre noticias y las funciones que se pueden realizar sobre el mismo.

Reportes de Afectaciones: Aquí se podrán consultar los reportes sobre cualquier afectación y las funciones que se pueden realizar sobre el mismo.

Afectaciones: Esta sección brindará información específica sobre las afectaciones que puedan ocurrir en el sistema.

Panel de Control: Esta opción llevará al usuario al panel de control del sistema de estación de bombeo, donde se lleva el monitoreo de las variables e medición y estado de las bombas.

Registros de Usuarios: En esta sección se podrá acceder a los registros de entrada y salida de los operadores del sistema.

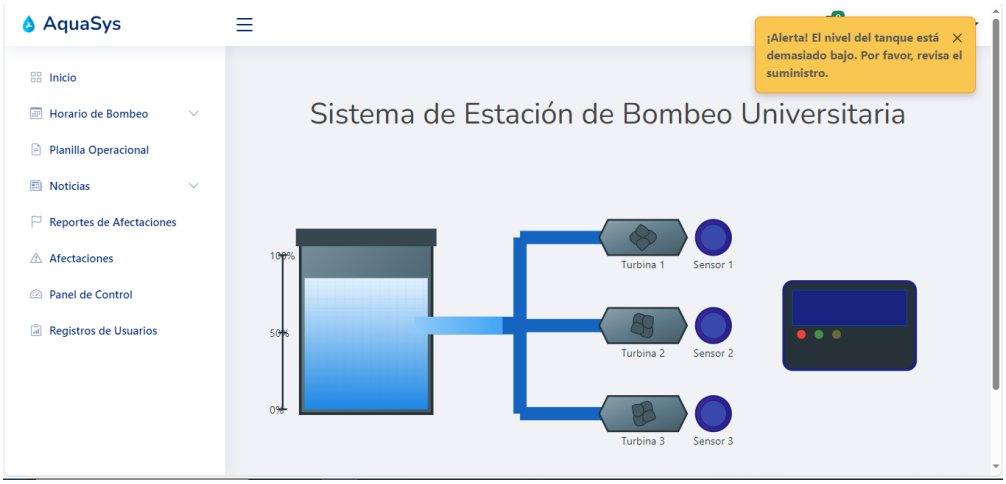


Figura 3.2. Menu del sistema. Elaboración propia

3.5. Diagrama de despliegue

El diagrama de despliegue es una herramienta fundamental en el diseño y documentación de sistemas de ingeniería complejos. Según Sommerville, 2011, este tipo de diagrama muestra la configuración de los nodos de hardware y los componentes de software que se ejecutan en ellos. Tal como indica Gamma E. Helm R. y V. J., 2022, los diagramas de despliegue ayudan a los desarrolladores y a los interesados a visualizar cómo se distribuye el sistema en un entorno de ejecución. Esta visibilidad facilita el análisis, la toma de decisiones

y la gestión eficaz del sistema de estación de bombeo durante su implementación y operación.

Como se muestra en la figura **??**, el diagrama de despliegue del sistema tiene como objetivo ilustrar la arquitectura física y la interconexión de los principales elementos que conforman esta infraestructura crítica. Esta representación gráfica es clave para comprender la distribución y las relaciones entre los componentes, como los sensores, el contolaor lógico programable, el broker MQTT, el servidor de base de datos, la aplicación y el cliente.

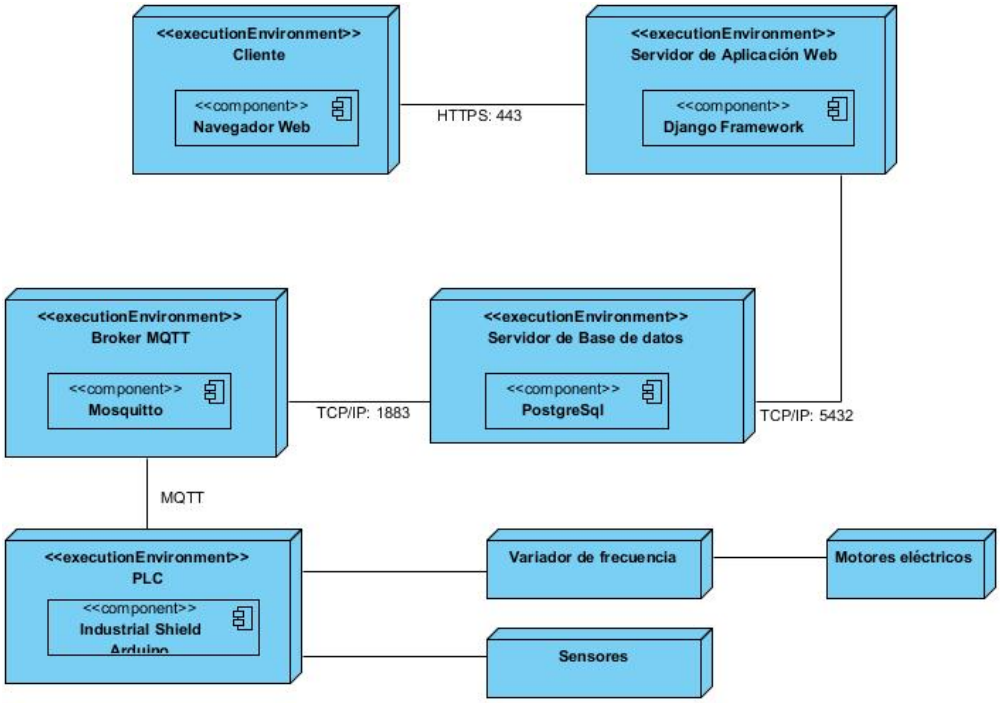


Figura 3.3. Diagrama de despliegue del sistema. Elaboración propia

3.6. Estrategia de prueba

El proceso de pruebas de software asegura que un sistema cumpla con los estándares de calidad esperados antes de su despliegue. La metodología adoptada para el proyecto, combina principios de la XP con pruebas de rendimiento.

Las pruebas se realizarán en todos los niveles, desde pruebas unitarias, desarrolladas en Django mediante TestCase en los archivos definidos , hasta pruebas de aceptación para validar requisitos específicos con los usuarios finales.

El uso de técnicas modernas y la consulta de estándares internacionales, como lo mencionan **Jamil2018** permitirán asegurar la calidad y robustez del sistema. Se priorizará la interacción del usuario final, evaluando

tanto funcionalidades básicas como aspectos no funcionales críticos.

Tabla 3.13. Estrategia de pruebas

Tipo de Prueba	Objetivo	Herramienta
Unitarias	Validar funcionalidades individuales del sistema	Herramientas integradas que ofrece el framework Django , utilizando el comando python manage.py test
Aceptación	Asegurar que el sistema satisface las necesidades y expectativas de los usuarios finales	Casos de pruebas
Rendimiento	Ayuda a identificar cuellos de botella, tiempos de respuesta y límites de capacidad de una aplicación	JMeter

3.7. Pruebas de Aceptación

Como se mencionó en el Capítulo 1, en la sección de herramientas a utilizar, la metodología XP propone la implementación de pruebas de aceptación para verificar el correcto funcionamiento de los requisitos establecidos por el cliente. Estas pruebas de aceptación se llevan a cabo en cada una de las iteraciones, lo que hace esencial obtener resultados satisfactorios para poder avanzar a la siguiente iteración. En las siguientes secciones, se presentarán las pruebas realizadas. Esta metodología asegura que cada etapa del desarrollo cumple con las expectativas del cliente antes de proceder a la siguiente, garantizando así la calidad y la eficacia del producto final. A continuación, se muestran algunas de las pruebas realizadas:

3.7.1. Iteración I

Tabla 3.14. Prueba de aceptación # 1

Caso de prueba de aceptación	
Código: HU2_P1	**Historia de usuario:** 2
Nombre: Modificación de Horario de Bombeo	
Descripción: Prueba para la funcionalidad: Modificar Horario de Bombeo	
Condiciones de ejecución: • El usuario debe estar autenticado en el sistema. • Debe existir un horario previo en el sistema.	

Continúa en la próxima página

Tabla 3.14. Continuación de la página anterior

Pasos de ejecución: • El usuario accede a la sección de horarios. • El usuario selecciona el horario a modificar. • El usuario modifica los campos Hora, Presión y Equipos. • El usuario guarda los cambios.
Resultados esperados: • El sistema valida los datos ingresados. • El sistema muestra un mensaje de confirmación. • El sistema actualiza el horario en la base de datos. • El sistema muestra el horario actualizado.

Tabla 3.15. Prueba de aceptación # 2

Caso de prueba de aceptación	
Código: HU 15_P 1	**Historia de usuario:** 15
Nombre: Autenticar Usuario	
Descripción: Prueba para la funcionalidad: Autenticar Usuarios	
Condiciones de ejecución: • El usuario debe tener una cuenta registrada en el sistema. • El usuario debe estar en la página de inicio de sesión.	
Pasos de ejecución: • El usuario ingresa su nombre de usuario. • El usuario ingresa su contraseña. • El usuario hace clic en Iniciar Sesión.	
Resultados esperados: • El sistema valida las credenciales. • El sistema permite el acceso si las credenciales son correctas. • El sistema muestra mensaje de error si las credenciales son incorrectas. • El sistema redirige al usuario a su página principal según su rol.	

3.7.2. Iteración II

Tabla 3.16. Prueba de aceptación # 3

Caso de prueba de aceptación	
Código: HU 10_P 1	**Historia de usuario:** 10

Continúa en la próxima página

Tabla 3.16. Continuación de la página anterior

Nombre: Configuración de Parámetros del Variador
Descripción: Prueba para la funcionalidad: Configuración del Variador de Frecuencia
Condiciones de ejecución: • El usuario debe estar autenticado con rol de Operador o Eléctrico. • El variador debe estar en línea y operativo.
Pasos de ejecución: • El usuario accede a la sección de configuración del variador. • El usuario ajusta el parámetro de presión de salida. • El sistema muestra mensaje de advertencia. • El usuario confirma los cambios.
Resultados esperados: • El sistema valida que los parámetros estén dentro de rangos seguros. • El sistema aplica los cambios al variador. • El sistema registra los cambios en el historial. • El sistema muestra la confirmación de cambios exitosos.

Tabla 3.17. Prueba de aceptación # 4

Caso de prueba de aceptación	
Código: HU 22_P 1	**Historia de usuario:** 22
Nombre: Registro Automático de Datos	
Descripción: Prueba para la funcionalidad: Registro automático de plantilla de control	
Condiciones de ejecución: • El sistema debe estar en funcionamiento • Los sensores y equipos deben estar operativos • Debe haber conexión con la base de datos	
Pasos de ejecución: • El sistema recopila datos de operación automáticamente • Se registran valores de presión, nivel del tanque, frecuencia, etc. • Se almacena la hora y fecha del registro • Se guardan los datos de los equipos en funcionamiento	
Resultados esperados: • Los datos se registran correctamente en la base de datos • Se incluyen todos los parámetros requeridos • El registro tiene marca de tiempo precisa • Los datos son consistentes con las lecturas de los sensores	

3.7.3. Iteración III

Tabla 3.18. Prueba de aceptación # 5

Caso de prueba de aceptación	
Código: HU 12_P 1	**Historia de usuario:** 12
Nombre: Creación de Nueva Afectación	
Descripción: Prueba para la funcionalidad: Crear de Afectaciones	
Condiciones de ejecución: • El usuario debe estar autenticado como Operador. • El usuario debe tener acceso al módulo de afectaciones.	
Pasos de ejecución: • El usuario accede al módulo de afectaciones. • El usuario selecciona Crear nueva afectación. • El usuario completa los campos: tipo, descripción, ubicación y estado. • El usuario guarda la afectación.	
Resultados esperados: • El sistema valida los campos ingresados. • El sistema registra la nueva afectación en la base de datos. • El sistema genera un identificador único para la afectación. • El sistema muestra un mensaje de confirmación.	

Tabla 3.19. Prueba de aceptación # 6

Caso de prueba de aceptación	
Código: HU1_P1	**Historia de usuario:** 1
Nombre: Restablecer Horario de Bombeo	
Descripción: Prueba para la funcionalidad: Restablecer Horario de Bombeo	
Condiciones de ejecución: • El usuario debe estar autenticado en el sistema. • Debe existir un horario modificado previamente.	
Pasos de ejecución: • El usuario accede a la sección de horarios. • El usuario visualiza el botón Restablecer. • El usuario hace clic en el botón Restablecer.	

Continúa en la próxima página

Tabla 3.19. Continuación de la página anterior

Resultados esperados: • El sistema muestra un mensaje de confirmación. • El sistema restablece el horario a su configuración predeterminada. • El sistema muestra el horario actualizado.

3.7.4. Iteración IV

Tabla 3.20. Prueba de aceptación # 7

Caso de prueba de aceptación	
Código: HU4_P1	**Historia de usuario:** 4
Nombre: Crear de Nueva Noticia	
Descripción: Prueba para la funcionalidad: Añadir Nueva Noticia	
Condiciones de ejecución: • El usuario debe estar autenticado en el sistema. • El usuario debe tener permisos de Eléctrico.	
Pasos de ejecución: • El usuario accede a la sección de noticias. • El usuario hace clic en Agregar noticia. • El usuario completa los campos Título, Contenido. • El usuario selecciona una imagen (opcional). • El usuario hace clic en Agregar.	
Resultados esperados: • El sistema valida los campos ingresados. • El sistema almacena la noticia en la base de datos. • El sistema muestra un mensaje de confirmación. • La nueva noticia aparece en la lista de noticias.	

Tabla 3.21. Resultado de las pruebas de aceptacion

Iteración	Total de pruebas	Cant. No Satisfactorias
1	8	0
2	7	3
3	8	2
4	10	0

A continuación se encuentran las no conformidades destacadas en el proceso de pruebas de aceptación, cabe mencionar que fueron solucionadas cada una de estas:

I. Acepta número en los campos a introducir letras.
II. Al modificar un elemento no notifica un mensaje de confirmación de éxito al usuario.
III. Permitir una contraseña de baja complejidad
IV. Errores de funcionamiento de algunas rutas: se arreglaron corrigiendo las direcciones de las vistas en los controladores.

3.8. Pruebas Unitarias

El desarrollo de software implica no solo la creación de funcionalidades, sino también la garantía de su correcto funcionamiento. En este contexto, las pruebas unitarias se constituyen como una herramienta fundamental para verificar la integridad y el comportamiento esperado de cada componente individual de un sistema de información. El presente capítulo aborda el proceso de validación de los modelos desarrollados para el sistema de gestión de bombeo, mediante la implementación de pruebas unitarias exhaustivas. El objetivo principal radica en:

Verificación Estructural : Comprobar que cada modelo se crea correctamente con los campos y restricciones definidos.
Validación de Comportamiento: Evaluar que los métodos y relaciones entre modelos funcionen según las especificaciones técnicas del sistema.
Aseguramiento de Integridad: Garantizar que los datos se almacenan, recuperan y manipulan de manera consistente y sin errores.

- Simplicidad: Cada prueba valida una única funcionalidad del modelo
- Retroalimentación: Las pruebas proporcionan resultados inmediatos sobre el funcionamiento del código
- Coraje: Las pruebas permiten realizar cambios con confianza
- Comunicación: Las pruebas actúan como documentación viva del sistema (M. L. y R. R., 2023)

3.8.1. Código de Pruebas Unitarias

```python
from django.test import TestCase
from django.contrib.auth import get_user_model
from bombeo.models import Registro

User  = get_user_model()

class RegistroModelTest(TestCase):
    def setUp(self):
        # Crear un usuario para asociar con el registro
        self.usuario = User.objects.create_user(username='testuser', password='testpass')

    def test_crear_registro_entrada(self):
        # Crear un registro de entrada
        registro = Registro.objects.create(usuario=self.usuario, accion='entrada')

        # Verificar que el registro se haya creado correctamente
        self.assertEqual(registro.usuario, self.usuario)
        self.assertEqual(registro.accion, 'entrada')
        self.assertIsNotNone(registro.fecha_hora)

    def test_crear_registro_salida(self):
        # Crear un registro de salida
        registro = Registro.objects.create(usuario=self.usuario, accion='salida')

        # Verificar que el registro se haya creado correctamente
        self.assertEqual(registro.usuario, self.usuario)
        self.assertEqual(registro.accion, 'salida')
        self.assertIsNotNone(registro.fecha_hora)

    def test_str_metodo(self):
        # Crear un registro
        registro = Registro.objects.create(usuario=self.usuario, accion='entrada')
```

Figura 3.4. Ejemplo de prueba unitaria

3.8.2. Notas sobre las Pruebas

Se realizaron un total de 45 pruebas unitarias realizadas en 2 iteraciones ,estas cubren todos los modelos del sistema de bombeo. Cada clase de prueba incluye verificaciones de:

- Creación básica de objetos
- Validación de campos
- Métodos específicos del modelo
- Representación en cadena (__str__)
- Casos límite y validaciones especiales

```
(env) D:\Programacion Web\1era practica\Mi tesis\GestionBombeo\gestion_agua>py manage.py test
Found 45 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....presion actual: 30
Bombas encendidas: 3
<QuerySet [<Bomba: Bomba: Bomba 1, Apagada>, <Bomba: Bomba: Bomba 2, Apagada>, <Bomba: Bomba: Bomba 3, Apagada>]>
.presion actual: 10
Bombas encendidas: 1
<QuerySet [<Bomba: Bomba: Bomba 1, Apagada>, <Bomba: Bomba: Bomba 2, Apagada>, <Bomba: Bomba: Bomba 3, Apagada>]>
.....
----------------------------------------------------------------------
Ran 45 tests in 27.908s

OK
Destroying test database for alias 'default'...
```

Figura 3.5. Resultado de las pruebas unitarias

3.9. Pruebas de rendimiento

El desarrollo de sistemas computacionales contemporáneos requiere una comprensión profunda más allá de la funcionalidad básica, situando las pruebas no funcionales como un dominio fundamental para garantizar la calidad y eficiencia de los sistemas de información. Dentro de este contexto, las pruebas de rendimiento emergen como una herramienta metodológica crítica para desentrañar el comportamiento sistémico bajo diversas condiciones operativas (Molyneaux, 2009).

Las pruebas no funcionales trascienden la mera verificación de requisitos operativos, constituyéndose como un mecanismo riguroso de evaluación que permite comprender la interacción dinámica entre componentes tecnológicos, recursos computacionales y demandas de procesamiento. Su relevancia científica radica en la capacidad de revelar aspectos latentes del desempeño que no son inmediatamente perceptibles mediante análisis funcionales tradicionales.

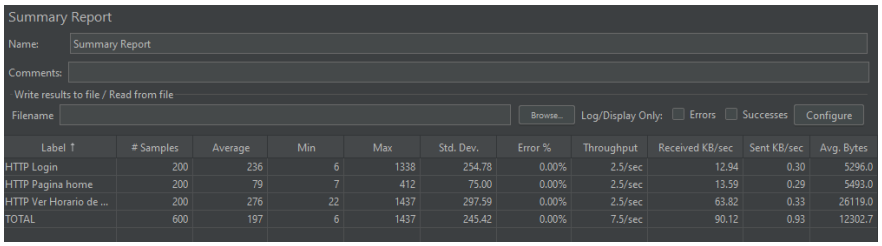
En el panorama de la ingeniería de software, las pruebas de rendimiento representan un vector analítico que permite caracterizar el comportamiento sistémico mediante la evaluación sistemática de parámetros como eficiencia, estabilidad, capacidad de respuesta y utilización de recursos. Esta aproximación metodológica implica una comprensión multidimensional que integra aspectos técnicos, algorítmicos y de arquitectura computacional.

La dimensión de las pruebas de rendimiento se despliega principalmente a través de dos modalidades analíticas fundamentales: las pruebas de carga y las pruebas de estrés. Cada una de estas modalidades contribuye de manera distintiva a la comprensión integral del comportamiento sistémico, permitiendo revelar características específicas del rendimiento computacional.

Las pruebas de carga emergen como un mecanismo de evaluación que simula escenarios operativos normalizados, posibilitando la caracterización del comportamiento del sistema bajo condiciones de uso previstas. Mediante la simulación de cargas de trabajo representativas, se pueden identificar patrones de respuesta, umbrales de saturación y puntos de potencial degradación del rendimiento.(Menascé y Almeida, 2002)

En contraste, las pruebas de estrés se configuran como un instrumento metodológico orientado a llevar los sistemas más allá de sus límites operativos convencionales. Esta modalidad de evaluación permite identificar mecanismos de resiliencia, puntos críticos de falla y capacidades de recuperación ante escenarios de sobrecarga, constituyéndose como un elemento fundamental para garantizar la robustez de arquitecturas computacionales(Smith y Williams, 2007).

A continuación se muestran las pruebas realizadas con la herramienta JMeter anteriormente mencionadas a las plantillas con mas concurrencia de usuarios :



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Login	200	236	6	1338	254.78	0.00%	2.5/sec	12.94	0.30	5296.0
HTTP Pagina home	200	79	7	412	75.00	0.00%	2.5/sec	13.59	0.29	5493.0
HTTP Ver Horario de ...	200	276	22	1437	297.59	0.00%	2.5/sec	63.82	0.33	26119.0
TOTAL	600	197	6	1437	245.42	0.00%	7.5/sec	90.12	0.93	12302.7

Figura 3.6. Vista de prueba de carga y estres

Luego de realizar estas pruebas con 300 usuario se mostraron los siguientes resultados y tiempo de respuesta:

- El sistema es capaz de manejar al menos 300 usuarios concurrentes sin degradación significativa del rendimiento.

- Los tiempos de respuesta para operaciones regulares no deben exceder los 4 segundos bajo condiciones normales de carga. Ver figura 3.7.

- Las actualizaciones en tiempo real de los datos de monitoreo (como niveles de tanques y estado de bombas) se refrescan cada 5 segundos o menos.

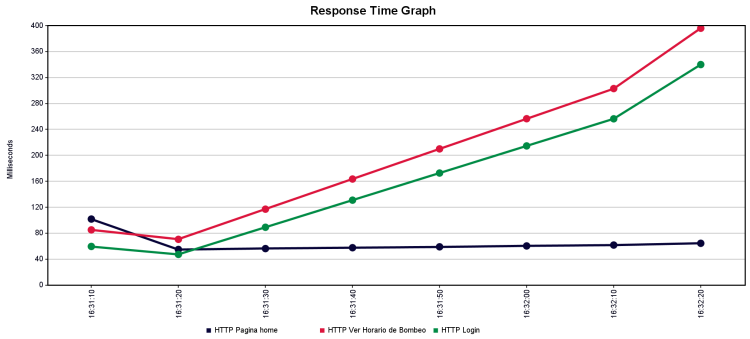


Figura 3.7. Tiempo de respuesta

3.10. Conclusiones del Capitulo 3.

1. El diagrama de despliegue logró representar la arquitectura física y la interconexión de los principales elementos que conforman la infraestructura crítica el sistema.

2. Se describieron un total de 33 tareas de ingeniería, delineando las acciones a seguir por el equipo de programación para desarrollar el sistema.

3. Las pruebas realizadas fueron fundamentales para mejorar la robustez y seguridad del sistema.

4. Durante las pruebas de aceptación se identificaron 5 inconformidades que se solucionaron de manera satisfactoria.

Conclusiones

1. Se identificaron las limitaciones del sistema manual de bombeo de agua en la UCI y se estableció una base teórica sólida para su automatización mediante herramientas como Django, MQTT y SQLite3, integrando conceptos de Iot e Industria 4.0.

2. El diseño basado en la metodología XP y el patrón MVT permitió una solución escalable y adaptable, alineada con los requisitos actuales y preparada para futuras mejoras.

3. La implementación iterativa validó el sistema como eficiente y fiable. Las pruebas confirmaron tiempos de respuesta óptimos, facilidad de uso y cumplimiento de los requisitos operativos.

4. El sistema desarrollado optimiza recursos, reduce errores y mejora la toma de decisiones en tiempo real. Su efectividad y replicabilidad lo posicionan como un modelo tecnológico para contextos similares.

Recomendaciones

- Migrar la base de Datos hacia un Servidor postgreSQL.
- Integrar la solución con el servidor NTFY de notificaciones a través de publicación/suscripción.
- Implementar una aplicación móvil cliente que consuma las notificaciones y alertas sobre el bombeo de agua.

Acrónimos

. 5

CRC Clase-Responsabilidad-Colaboración. 29

GALBA Guardian del alba. 1, 5
GOF Gang of Four. 31
GRASP Patrones de software de asignación de responsabilidades generales. 30

Iot Internet de las cosas. 1, 6, 15
ISPJAE Instituto Superior Politécnico José Antonio Echeverría. 6

MQTT Transporte de telemetría de Message Queue Server. 15, 16
MVC Modelo-vista-controlador. 12
MVT Modelo-vista-plantilla. 12

SCADA Supervisión, Control y Adquisición de Datos. 1, 5

UCI Universidad de las Ciencias Informáticas. 1–5, 7, 8, 33–47
UNAM Universidad Nacional Autónoma de México. 5

VSCode Codigo de Estudio Visual. 11

XP Programación extrema. 7, 8, 24, 27, 42

Referencias bibliográficas

A, Banks y R., Gupta, 2023. *MQTT Version 5.0 with Errata. OASIS Open.* OASIS Open. (vid. pág. 14).

A., Holovaty y J., Kaplan-Moss, 2009. *The Definitive Guide to Django: Web Development Done Right.* Ed. por APRESS. (vid. pág. 12).

A., Kumar y R., Bhatia, 2021. User Stories: An Empirical Study of Practice and Challenges. *Journal of Software: Evolution and Process.* (Vid. pág. 27).

AL, Gómez Cabrera et, 2020. Mejoras en el desarrollo de despliegues con el SCADA GALBA (vid. pág. 1).

B., Coelho y A., Andrade Campos, 2022. Energy efficiency in water pumping systems: An integrated approach. *Journal of Cleaner Production*. Vol. 330. Disponible desde DOI: 10.1016/j.jclepro.2021.129751 (vid. pág. 4).

BASS, L. Clements P. y R., Kazman, 2021. *Software Architecture in Practice 4th ed.* Addison Wesley Professional. (vid. pág. 16).

DAYLEY, B., 2021. *Django for Professionals: Production Websites with Python and Django.* WelderMedia Publishing. (vid. pág. 32).

DUSUNIOT, 2024. Cómo funciona la recopilación de datos de IoT [Guía completa 2024]. Url: https://www.dusuniot.com/es/blog/iot-data-collection/ (vid. pág. 8).

F., Dalpiaz y M., Parente, 2023. Quality Assessment of User Stories in Agile Projects. *Requirements Engineering Journal.* (Vid. pág. 26).

F, Silberschatz A. Korth H. y S, Sudarshan, 2021. *Database System Concepts 7th ed.* McGraw Hill Education. (vid. pág. 13).

FOUNDATION, Django Software, 2023. Django documentation. Url: https://docs.djangoproject.com/ (vid. pág. 12).

FOWLER, M., 2023. *Patterns of Enterprise Application Architecture 2nd ed.* Addison Wesley Professional. (vid. pág. 16).

GAMMA E. HELM R., Johnson R. y J., Vlissides, 2022. *Design Patterns: Elements of Reusable Object Oriented Software ,Anniversary ed.* Addison Wesley Professional. (vid. págs. 17, 43).

GARCÍA GARCÍA, J.A. et al., 2019. *A Model Driven Approach for User Story Requirements Specification in Agile Development.* Information y Software Technology. (vid. pág. 26).

GARCÍA RUIZ, M., 2021. Sistemas de automatización industrial: evolución y tendencias actuales. *Revista de Ingeniería Industrial*. Vol. 15, n.º 2, págs. 45-62 (vid. pág. 1).

GARCIA, R. y FERNANDEZ, P., 2020. Comparative Analysis of Performance Testing Tools. En: *Comparative Analysis of Performance Testing Tools. 2020 International Conference on Software Engineering*, págs. 112-125 (vid. pág. 17).

GEORGE, J., 2023. *Django Design Patterns and Best Practices 4th ed.* Packt Publishing. (vid. pág. 32).

HIVEMQ., 2022. *MQTT Essentials: A Comprehensive Guide to MQTT Protocol 2nd ed.* HiveMQ GmbH. (vid. pág. 15).

J., Date C., 2019. *Database Design and Relational Theory: Normal Forms and All That Jazz*. 2nd ed. Apress. (vid. pág. 13).

J., Gope D. Schlais D. y H., Lipasti M., 2022. Architectural Support for Server-Side PHP Processing. *IEEE Computer Architecture Letters*. Vol. 21, n.º 1, págs. 5-8 (vid. pág. 11).

J., Rodríguez A. Martínez y C., López, 2017. Mejoras en el desarrollo de despliegues con el SCADA GALBA. *Revista de Automatización y Control*. Vol. 8, n.º 1, págs. 78-93 (vid. pág. 1).

JOURNAL OF SYSTEMS, Rahman M. et al.. y SOFTWARE., 2023. User Stories in Agile Development: A Systematic Literature Review. *Journal of Systems and Software.* (Vid. pág. 26).

K., Beck y ANDRES, 2004. *Extreme Programming Explained: Embrace Change*. 2nd. Addison-Wesley Professional (vid. págs. 9, 18, 28).

K., Schwaber y SUTHERLAND, J., 2020. *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game.* Scrum.org. (vid. págs. 10, 19).

KUMAR, A. y SHARMA, R., 2021. Performance Evaluation Techniques Using Apache JMeter. *Journal of Software Engineering and Testing*. Vol. 15, n.º 3, págs. 45-59 (vid. pág. 17).

L., Martínez y R., Rodríguez, 2023. Aplicación de XP en sistemas de control: Casos de estudio en América Latina. *Revista Cubana de Ingeniería,* vol. 44, n.º 2, págs. 45-58 (vid. págs. 10, 19, 50).

L., Rodríguez M. García y J., Torres, 2019. Aplicación de patrones GRASP en el desarrollo de software. *Revista de Ingeniería de Software*. Vol. 15, n.º 2, págs. 123-136. (Vid. págs. 10, 19).

LARRAÑAGA, Luis Alfredo, 2024. La Sinergia entre Inteligencia Artificial (IA) e Internet de las Cosas (IoT): Un Futuro Prometedor. Url: `https://foqum.io/blog/la-sinergia-entre-inteligencia-artificial-ia-e-internet-de-las-cosas-iot-un-futuro-prometedor/` (vid. pág. 8).

LUTZ, M., 2013. *Learning Python*. 5th ed. Ed. por MEDIA., OReilly (vid. pág. 12).

M., Cohn, 2004. *User Stories Applied: For Agile Software Development.* Addison Wesley Professional. (vid. págs. 9, 18).

M., Fowler y J., Highsmith, 2011. The Agile Manifesto. *Software Development Magazine*. Vol. 9, n.º 8, págs. 28-35 (vid. pág. 28).

M., Pérez Sánchez M. Sánchez Romero F. J. Ramos H. y A., López Jiménez P., 2020. Smart water grids with energy recovery: Analysis and a case study. Journal of Water Resources Planning and Management, vol. 146, n.º 3, págs. 05020002. (Vid. pág. 5).

M., Richards, 2021. *Fundamentals of Software Architecture.* OReilly Media. (vid. pág. 16).

MAKAI, M., 2023. *Full Stack Python: Security Guide.* Full Stack Python. (vid. pág. 32).

MARTINEZ, S. y NAVARRO, L., 2021. Taxonomía de Herramientas de Pruebas de Carga en Entornos Computacionales. *Revista Iberoamericana de Tecnologías de la Información*. Vol. 15, n.º 3, págs. 22-38 (vid. pág. 17).

MENASCÉ, Daniel A. y ALMEIDA, Virgilio A.F., 2002. Performance by Design: Computer Capacity Planning by Example (vid. pág. 52).

MICROSOFT, 2023. Visual Studio Code Documentation. Url: `https://code.visualstud` (vid. pág. 13).

MOHAN, R. y PATEL, S., 2020. Comprehensive Analysis of Load Testing Techniques Using Open-Source Tools. *International Journal of Computer Science and Information Technology*. Vol. 12, n.º 4, págs. 67-81 (vid. pág. 18).

MOLYNEAUX, Ian, 2009. *The Art of Application Performance Testing*. Sebastopol, CA: OReilly Media (vid. pág. 52).

MOMJIAN, B., 2021. *PostgreSQL: Introduction and Concepts.* Addison Wesley Professional. (vid. pág. 13).

NEWMAN, S., 2021. *Building Microservices 2nd ed.* OReilly Media. (vid. pág. 16).

P., González Bravo R. Martínez Santos y A., Ramírez, 2022. Gestión integral de sistemas de bombeo de agua: Manual técnico y operativo. *Revista Ingeniería Hidráulica,* vol. 43, n.º 2, págs. 78-92. Disponible desde DOI: `10.15446/ing.hidraul.v43n2.89234` (vid. págs. 5, 7).

P., Kumar S. Singh y M., Kumar, 2022. Evolution of Software Development Methodologies: A Sys-tematic Review. *Journal of Software Engineering and Applications,* vol. 15, n.º 4, págs. 156-172. (Vid. págs. 9, 18).

R., Molina, 2022. Desarrollo e implementación de sistemas SCADA en Cuba: Casos de estudio. *Revista Cubana de Ingeniería*. Vol. 43, n.º 1, págs. 12-25. (Vid. pág. 1).

RAIMUNDO CARLOS, Ramos Guardarrama Josnier Hernandez Areu Orestes Silverio Freire y, 2019. Sistema de supervisión para el monitoreo de redes eléctricas inteligentes. *Ingeniería Energética*. Vol. 40, págs. 264-272. ISSN 1815-5901. Url: `http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1815-59012019000300264&nrm=iso` (vid. pág. 8).

RAVINDRAN, A., 2022. *Django 4 By Example: Build powerful and reliable Python web applications from scratch*. 4th ed. Packt Publishing. (vid. pág. 32).

RODRIGUEZ CAPOTE, Letsi de la Caridad; HERNANDEZ BARRIENTO, Rosaly; CORRALES DIAZ, Dayrien; GOMEZ GARCIA, Carlos y FERNANDEZ INFANTE, Carlos Alberto, 2016. *Almacen de Datos para el SCADA GALBA*. B.S. thesis (vid. pág. 7).

RODRIGUEZ, M. y LOPEZ, J., 2018. Comprehensive Overview of Performance Testing Methodologies. *International Journal of Computer Science*. Vol. 9, n.º 2, págs. 45-60 (vid. pág. 17).

S., Pressman R. y R., Maxim B., 2015. *Software Engineering: A Practitioners Approach*. 8th ed. McGraw-Hill Education. (vid. págs. 11, 28).

SEBESTA, R. W., 2016. *Concepts of Programming Languages 11th ed.* Ed. por PEARSON. (vid. pág. 12).

SMITH, Connie U. y WILLIAMS, Lloyd G., 2007. Performance Engineering of Software Systems. En: *Performance Engineering of Software Systems*. *International Conference on Software Engineering*, págs. 697-698 (vid. pág. 52).

SOMMERVILLE, I., 2011. *Software Engineering 9th ed.* Addison-Wesley (vid. págs. 9, 13, 18, 31, 43).

SWITNICKA, Karolina; SUCHORAB, Pawe y KOWALSKA, Beata, 2017. The optimisation of a water distribution system using Bentley WaterGEMS software. En: *The optimisation of a water distribution system using Bentley WaterGEMS software*. *ITM Web of Conferences*. Vol. 15, pág. 03009 (vid. pág. 7).

TOLL, W., 2023. *Python and MQTT: Building IoT Solutions 3rd ed.* Manning Publications. (vid. pág. 15).

V., Vernon, 2022. *Implementing Domain Driven Design 2nd ed.* Addison Wesley Professional. (vid. pág. 16).

WAHER, P., 2021. *IoT Protocols and Communications Standards.* Packt Publishing. (vid. pág. 15).

WELLS, D., 2013. *Extreme Programming: A Gentle Introduction.* Extreme Programming.org. (vid. pág. 28).

X., Wang Y. Liu y J., Chen, 2023. Agile Methodologies in Control Systems Development: A Contemporary Analysis. *International Journal of Software Engineering,* vol. 12, n.º 3, págs. 234-249 (vid. págs. 10, 19).

Generado con LaTeX: 8 de diciembre de 2024: 10:24am

Apéndices

APÉNDICE A

Anexos

A.1. Tarjetas CRC

Tabla A.1. Tarjeta CRC # 3

Tarjeta CRC	
Clase: UserRelation	
Responsabilidad	**Colaboración**
• Gestionar relaciones de amistad entre usuarios • Rastrear solicitudes de amistad pendientes • Controlar el estado de aceptación de relaciones • Permitir la conexión entre usuarios del sistema	• Usuario (modelo de usuario principal) • Mensajes (posible integración para comunicación)

Tabla A.2. Tarjeta CRC # 4

Tarjeta CRC	
Clase: Messages	
Responsabilidad	**Colaboración**

Continúa en la próxima página

Tabla A.2. Continuación de la página anterior

• Almacenar mensajes entre usuarios • Registrar el remitente y destinatario de cada mensaje • Marcar mensajes como leídos o no leídos • Mantener un registro cronológico de comunicaciones • Gestionar la marca de tiempo de los mensajes	• Usuario (para identificar remitente y destinatario) • UserRelation (posible filtro de comunicaciones)

Tabla A.3. Tarjeta CRC # 5

Tarjeta CRC	
Clase: Registro	
Responsabilidad	**Colaboración**
• Registrar entradas y salidas de usuarios • Mantener un histórico de actividad de usuarios • Asociar acciones a usuarios específicos • Proporcionar trazabilidad de accesos al sistema	• Usuario (para asociar registros) • Sistema de autenticación

Tabla A.4. Tarjeta CRC # 6

Tarjeta CRC	
Clase: Bomba	
Responsabilidad	**Colaboración**
• Representar el estado de las bombas (encendido/apagado) • Identificar bombas individuales en el sistema • Permitir control y monitoreo de infraestructura hidráulica • Gestionar la activación y desactivación de bombas	• Variador (ajuste de presión y control) • HorarioBombeo (definición de intervalos) • RegistroOperacional (registro de estado)

Tabla A.5. Tarjeta CRC # 7

Tarjeta CRC	
Clase: PresionEntrada	
Responsabilidad	**Colaboración**
• Registrar la presión de entrada al sistema • Almacenar valores de presión con marca temporal • Monitorear condiciones de entrada de fluido • Proporcionar datos para análisis de rendimiento	• NivelTanque (ajuste de nivel) • Variador (control de presión) • RegistroOperacional (logging)

Tabla A.6. Tarjeta CRC # 8

Tarjeta CRC	
Clase: Variador	
Responsabilidad	**Colaboración**
• Ajustar la presión de salida del sistema • Controlar la frecuencia de operación • Gestionar el estado de las bombas • Registrar cambios de presión y frecuencia	• Bomba (control de estado) • PresionEntrada (comparación de presiones) • NivelTanque (influencia en nivel) • RegistroOperacional (registro de operaciones)

Tabla A.7. Tarjeta CRC # 9

Tarjeta CRC	
Clase: NivelTanque	
Responsabilidad	**Colaboración**
• Simular y registrar el nivel de llenado del tanque • Calcular cambios de nivel basados en presiones • Mantener un histórico de niveles de tanque • Gestionar el estado de llenado entre 0	• PresionEntrada (influencia en nivel) • Variador (ajuste de presión) • RegistroOperacional (logging)

Tabla A.8. Tarjeta CRC # 10

Tarjeta CRC	
Clase: VolumenBombeado	
Responsabilidad	**Colaboración**
• Registrar el volumen de agua bombeada • Almacenar datos de bombeo con marca temporal • Proporcionar métricas de producción • Facilitar la planilla operacional	• RegistroOperacional (integración de datos) • PlanillaHistorial (registro histórico)

Tabla A.9. Tarjeta CRC # 11

Tarjeta CRC	
Clase: Noticia	
Responsabilidad	**Colaboración**
• Publicar noticias en el sistema • Gestionar contenido informativo • Almacenar títulos, contenido e imágenes • Mantener un orden cronológico de publicaciones	• Usuario (posible restricción de publicación) • Sistema de gestión de contenidos

Tabla A.10. Tarjeta CRC # 12

Tarjeta CRC	
Clase: Reporte	
Responsabilidad	**Colaboración**
• Registrar afectaciones o incidencias • Documentar problemas en áreas específicas • Clasificar estado de resolución de reportes • Mantener un histórico de incidencias	• Usuario (creación y seguimiento) • Sistema de gestión de mantenimiento

Tabla A.11. Tarjeta CRC # 13

Tarjeta CRC
Clase: RegistroOperacional

Continúa en la próxima página

Tabla A.11. Continuación de la página anterior

Responsabilidad	**Colaboración**
• Consolidar datos de operación hidráulica • Registrar información horaria del sistema • Almacenar métricas de presión, nivel y volumen • Proporcionar base para análisis de rendimiento	• Variador • Bomba • NivelTanque • PresionEntrada • PlanillaHistorial

Tabla A.12. Tarjeta CRC # 14

Tarjeta CRC	
Clase: PlanillaHistorial	
Responsabilidad	**Colaboración**
• Agrupar registros operacionales por fecha • Mantener un historial consolidado de operaciones • Facilitar la revisión y análisis de datos históricos • Proporcionar una vista agregada de rendimiento	• RegistroOperacional • Reportes y análisis de sistema

A.2. Historias de usuario

Tabla A.13. Historia de usuario # 5

Historia de usuario	
Número: 5	**Nombre:** Modificar Horario
Usuario: Eléctrico	
Prioridad en negocio: Alto	**Riesgo en desarrollo:** Media
Puntos estimados: 0.5	**Iteración asignada:** 3
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permite al usuario seleccionar una franja del horario de bombeo y modificar sus datos de hora inicio, hora fin y presión, así como adicionar o eliminar franjas horarias.	
Observaciones: Los cambios deben respetar los parámetros operativos del sistema.	

Tabla A.14. Historia de usuario # 6

Historia de usuario	
Número: 6	**Nombre:** Restablecer Horario
Usuario: Eléctrico	
Prioridad en negocio: Alto	**Riesgo en desarrollo:** Media
Puntos estimados: 0.4	**Iteración asignada:** 3
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permite al usuario restablecer el horario de bombeo a la configuración predeterminada del sistema.	
Observaciones: La acción de restablecimiento debe ser trazable.	

Tabla A.15. Historia de usuario # 7

Historia de usuario	
Número: 7	**Nombre:** Visualizar Horario
Usuario: Cliente, Operador,Eléctrico	
Prioridad en negocio: Alto	**Riesgo en desarrollo:** Baja
Puntos estimados: 0.2	**Iteración asignada:** 3
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permite visualizar el horario de bombeo programado con sus respectivos datos.	
Observaciones: En caso de no haber modificaciones se muestra el horario predeterminado.	

Tabla A.16. Historia de usuario # 8

Historia de usuario	
Número: 8	**Nombre:** Modificar Noticia
Usuario: Eléctrico	
Prioridad en negocio: Medio	**Riesgo en desarrollo:** Baja
Puntos estimados: 0.3	**Iteración asignada:** 4
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permite seleccionar y modificar los datos de una noticia existente: Título, Contenido e Imagen.	
Observaciones: Mantener un registro de la última modificación.	

Tabla A.17. Historia de usuario # 9

Historia de usuario	
Número: 9	**Nombre:** Eliminar Noticia
Usuario: Eléctrico	
Prioridad en negocio: Medio	**Riesgo en desarrollo:** Baja
Puntos estimados: 0.3	**Iteración asignada:** 4
Programador responsable: Carlos Adrián Burley Ross	

Continúa en la próxima página

Tabla A.17. Continuación de la página anterior

Descripción: Permite seleccionar y eliminar una noticia del sistema.
Observaciones: La eliminación debe ser una acción confirmada por el usuario.

Tabla A.18. Historia de usuario # 10

Historia de usuario	
Número: 10	**Nombre:** Visualizar Noticia
Usuario: Cliente, Operador,Eléctrico	
Prioridad en negocio: Alto	**Riesgo en desarrollo:** Media
Puntos estimados: 0.4	**Iteración asignada:** 4
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permite mostrar las noticias y ver la información específica de cada una.	
Observaciones: Presentar las noticias de manera clara y ordenada.	

Tabla A.19. Historia de usuario # 11

Historia de usuario	
Número: 11	**Nombre:** Monitorear Estado de Bombas
Usuario: Operador,Eléctrico	
Prioridad en negocio: Alto	**Riesgo en desarrollo:** Media
Puntos estimados: 0.3	**Iteración asignada:** 2
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permite mostrar en tiempo real el estado actual de funcionamiento de las bombas de agua.	
Observaciones: Incluir detalles de rendimiento y estado operativo.	

Tabla A.20. Historia de usuario # 12

Historia de usuario	
Número: 12	**Nombre:** Descargar Reporte de Afectaciones
Usuario: Operador,Eléctrico	
Prioridad en negocio: Alto	**Riesgo en desarrollo:** Alta
Puntos estimados: 0.3	**Iteración asignada:** 3
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permite generar y descargar reportes sobre las afectaciones al abastecimiento de agua.	
Observaciones: El reporte debe ser detallado y en formato descargable.	

Tabla A.21. Historia de usuario # 13

Historia de usuario	
Número: 13	**Nombre:** Crear Reporte de Afectación
Usuario: Operador	
Prioridad en negocio: Alto	**Riesgo en desarrollo:** Media
Puntos estimados: 0.4	**Iteración asignada:** 3
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permite registrar un nuevo reporte de afectación con campos de Nombre, Área, Causa y Estado.	
Observaciones: Registro detallado y trazable de la afectación.	

Tabla A.22. Historia de usuario # 14

Historia de usuario	
Número: 14	**Nombre:** Modificar Reporte de Afectación
Usuario: Operador	
Prioridad en negocio: Alto	**Riesgo en desarrollo:** Media
Puntos estimados: 0.3	**Iteración asignada:** 3
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permite seleccionar y modificar los datos de un reporte de afectación existente.	
Observaciones: Mantener historial de modificaciones.	

Tabla A.23. Historia de usuario # 15

Historia de usuario	
Número: 15	**Nombre:** Eliminar Reporte de Afectación
Usuario: Operador	
Prioridad en negocio: Alto	**Riesgo en desarrollo:** Media
Puntos estimados: 0.3	**Iteración asignada:** 3
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permite seleccionar y eliminar un reporte de afectación del sistema.	
Observaciones: Eliminar solo con confirmación y registro de la acción.	

Tabla A.24. Historia de usuario # 16

Historia de usuario	
Número: 16	**Nombre:** Autenticar Usuarios
Usuario: Aministrador,Operador,Eléctrico	
Prioridad en negocio: Alto	**Riesgo en desarrollo:** Alta
Puntos estimados: 0.3	**Iteración asignada:** 3
Programador responsable: Carlos Adrián Burley Ross	

Continúa en la próxima página

Tabla A.24. Continuación de la página anterior

Descripción: Permite a los usuarios autenticarse en el sistema mediante Usuario y Contraseña.
Observaciones: Implementar medidas de seguridad robustas.

Tabla A.25. Historia de usuario # 17

Historia de usuario	
Número: 17	**Nombre:** Añadir Usuario
Usuario: Administrador	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 0.4	**Iteración asignada:** 1
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permite al administrador crear un nuevo usuario proporcionando los siguientes datos: Nombre de Usuario, Nombre Completo, Correo Electrónico, Rol y Contraseña.	
Observaciones: El administrador debe tener permisos especiales para crear nuevos usuarios en el sistema.	

Tabla A.26. Historia de usuario # 18

Historia de usuario	
Número: 18	**Nombre:** Modificar Usuario
Usuario: Administrador	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 0.3	**Iteración asignada:** 1
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permite al administrador seleccionar un usuario existente y modificar sus datos: Nombre de Usuario, Nombre Completo, Correo Electrónico y Rol.	
Observaciones: No se podrá modificar la contraseña en esta funcionalidad. Existe una historia de usuario separada para cambiar contraseña.	

Tabla A.27. Historia de usuario # 19

Historia de usuario	
Número: 19	**Nombre:** Eliminar Usuario
Usuario: Administrador	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 0.3	**Iteración asignada:** 1
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permite al administrador seleccionar uno o varios usuarios y eliminarlos del sistema.	
Observaciones: La eliminación de usuarios debe realizarse con precaución, ya que puede afectar el registro histórico de operaciones.	

Tabla A.28. Historia de usuario # 20

Historia de usuario	
Número: 20	**Nombre:** Monitorear Presiones de Agua
Usuario: Operador,Eléctrico	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 0.3	**Iteración asignada:** 2
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Mostrar y registrar en tiempo real la presión de salida y de entrada del agua en el tanque de la Estación a través de una interfaz gráfica.	
Observaciones: Las presiones deben mostrarse de forma clara y actualizada constantemente para permitir una supervisión eficiente.	

Tabla A.29. Historia de usuario # 21

Historia de usuario	
Número: 21	**Nombre:** Realizar Alertas Automáticas
Usuario: Sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 0.4	**Iteración asignada:** 2
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Generar y mostrar alertas automáticas cuando se detecten niveles críticos en el nivel de llenado del tanque.	
Observaciones: Las alertas deben ser visualmente distintivas y proporcionar información clara sobre el estado crítico del tanque.	

Tabla A.30. Historia de usuario # 22

Historia de usuario	
Número: 22	**Nombre:** Descargar Planilla de Operaciones
Usuario: Operador,Eléctrico	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 0.3	**Iteración asignada:** 3
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permitir exportar las planillas de control de operaciones en formato PDF a través de una interfaz gráfica.	
Observaciones: El documento exportado debe contener todos los detalles relevantes de las operaciones realizadas.	

Tabla A.31. Historia de usuario # 23

Historia de usuario	
Número: 23	**Nombre:** Registrar Automáticamente Registros Operacionales
Usuario: Sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 0.6	**Iteración asignada:** 2
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Registrar automáticamente los datos de operación, incluyendo: Hora, Presión de entrada, Equipos trabajando, Presión de salida, Nivel del tanque, Frecuencia de Variador, Volumen bombeado y Observaciones.	
Observaciones: El registro debe ser preciso y realizarse en tiempo real sin intervención manual.	

Tabla A.32. Historia de usuario # 24

Historia de usuario	
Número: 24	**Nombre:** Filtrar Registros Operacionales
Usuario: Operador,Eléctrico	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 0.4	**Iteración asignada:** 2
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permitir filtrar los registros operacionales por fecha y mostrar los datos registrados en ese día.	
Observaciones: La interfaz de filtrado debe ser intuitiva y proporcionar resultados rápidos y precisos.	

Tabla A.33. Historia de usuario # 25

Historia de usuario	
Número: 25	**Nombre:** Eliminar Registros Operacionales
Usuario: Administrador,Eléctrico	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimados: 0.4	**Iteración asignada:** 4
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permitir eliminar los registros operacionales del día previamente filtrado a través de una interfaz gráfica.	
Observaciones: Se debe solicitar confirmación antes de eliminar los registros para evitar pérdidas accidentales de información.	

Tabla A.34. Historia de usuario # 26

Historia de usuario	
Número: 26	**Nombre:** Enviar Mensajes en Tiempo Real
Usuario: Aministrador,Operador,Eléctrico	

Continúa en la próxima página

Tabla A.34. Continuación de la página anterior

Prioridad en negocio: Media	**Riesgo en desarrollo:** Baja
Puntos estimados: 0.5	**Iteración asignada:** 4
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permitir responder y escribir mensajes en tiempo real entre los usuarios a través de una interfaz gráfica.	
Observaciones: La interfaz de mensajería debe ser intuitiva y permitir una comunicación rápida entre usuarios.	

Tabla A.35. Historia de usuario # 27

Historia de usuario	
Número: 27	**Nombre:** Buscar Usuario
Usuario: Aministrador,Operador,Eléctrico	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 0.4	**Iteración asignada:** 1
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permitir buscar un usuario existente en el sistema a través de una interfaz gráfica.	
Observaciones: La búsqueda debe poder realizarse mediante diferentes criterios como nombre de usuario, nombre completo o rol.	

Tabla A.36. Historia de usuario # 28

Historia de usuario	
Número: 28	**Nombre:** Enviar Solicitud
Usuario: Aministrador,Operador,Eléctrico	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 0.3	**Iteración asignada:** 4
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permitir enviar una solicitud al usuario seleccionado para añadirlo a los contactos.	
Observaciones: El sistema debe notificar al usuario receptor de la solicitud de contacto.	

Tabla A.37. Historia de usuario # 29

Historia de usuario	
Número: 29	**Nombre:** Aceptar Solicitud
Usuario: Aministrador,Operador,Eléctrico	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 0.3	**Iteración asignada:** 4
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permitir aceptar las solicitudes de contacto enviadas por otros usuarios.	

Continúa en la próxima página

Tabla A.37. Continuación de la página anterior

Observaciones: Una vez aceptada la solicitud, el usuario debe aparecer en la lista de contactos.

Tabla A.38. Historia de usuario # 30

Historia de usuario	
Número: 30	**Nombre:** Registro Automático de Entrada y Salida
Usuario: Sistema	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Baja
Puntos estimados: 0.4	**Iteración asignada:** 1
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Registrar automáticamente la entrada y salida de usuarios con los campos: Nombre de Usuario, Rol, Acción y Fecha.	
Observaciones: El registro debe ser transparente para el usuario y realizarse de manera automática al iniciar o cerrar sesión.	

Tabla A.39. Historia de usuario # 31

Historia de usuario	
Número: 31	**Nombre:** Visualizar Registro de Control de Acceso
Usuario: Eléctrico	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 0.4	**Iteración asignada:** 4
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Mostrar mediante una interfaz gráfica las entradas y salidas de los operadores en los turnos diariamente.	
Observaciones: El registro debe permitir filtrar por fecha, usuario o rol para facilitar la revisión.	

Tabla A.40. Historia de usuario # 32

Historia de usuario	
Número: 32	**Nombre:** Visualizar Perfil de Usuario
Usuario: Aministrador,Operador,Eléctrico	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 0.2	**Iteración asignada:** 1
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Mostrar mediante una interfaz gráfica la información del usuario: Nombre de Usuario, Nombre Completo, Correo Electrónico y Rol.	
Observaciones: El perfil debe mostrar la información de manera clara y ordenada.	

Tabla A.41. Historia de usuario # 33

Historia de usuario	
Número: 33	**Nombre:** Cambiar Contraseña
Usuario: Aministrador,Operador,Eléctrico	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 0.3	**Iteración asignada:** 1
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Permitir modificar la contraseña mediante una interfaz gráfica, solicitando: contraseña actual, nueva contraseña y confirmación de nueva contraseña.	
Observaciones: Se deben implementar validaciones de seguridad como longitud mínima, complejidad de contraseña y coincidencia de nueva contraseña.	

A.3. Tareas

Tabla A.42. Tarea de desarrollo # 9

Tarea	
Número de tarea: 9	**Número de Historia de usuario:** 13
Nombre de la tarea: Modificar Horario	
Tipo de tarea: Funcionalidad	**Puntos estimados:** 0.4
Fecha de inicio: 4 de septiembre de 2024	**Fecha de fin:** 8 de septiembre de 2024
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Para permitir al usuario actualizar los horarios de bombeo existentes, se deben crear las rutas de acceso mediante el enrutador de Django. La vista debe capturar los campos "Hora", "Presiónz .Equiposz actualizarlos en la base de datos. La plantilla a utilizar debe permitir cargar el formulario creado desde Python y validar los campos según los datos definidos.	

Tabla A.43. Tarea de desarrollo # 10

Tarea	
Número de tarea: 10	**Número de Historia de usuario:** 15
Nombre de la tarea: Visualizar Horario	
Tipo de tarea: Funcionalidad	**Puntos estimados:** 0.3
Fecha de inicio: 8 de septiembre de 2024	**Fecha de fin:** 11 de septiembre de 2024
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Para permitir al usuario ver el horario de bombeo actual, se deben crear las rutas de acceso mediante el enrutador de Django. La vista debe recuperar los datos del horario de la base de datos y mostrarlos en una interfaz gráfica.	

Tabla A.44. Tarea de desarrollo # 11

Tarea	
Número de tarea: 11	**Número de Historia de usuario:** 22
Nombre de la tarea: Registrar automáticamente plantilla de control de operación	
Tipo de tarea: Funcionalidad	**Puntos estimados:** 3
Fecha de inicio: 6 de octubre de 2024	**Fecha de fin:** 27 de octubre de 2024
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Para permitir el registro automático de los datos de operación, se deben crear las rutas de acceso mediante el enrutador de Django. La vista debe capturar y registrar en la base de datos los datos como hora, presiones de entrada/salida, equipos en funcionamiento, nivel del tanque, frecuencia del variador, volumen bombeado y observaciones, sin requerir la intervención manual del usuario.	

Tabla A.45. Tarea de desarrollo # 12

Tarea	
Número de tarea: 12	**Número de Historia de usuario:** 26
Nombre de la tarea: Modificar Noticia	
Tipo de tarea: Funcionalidad	**Puntos estimados:** 0.3
Fecha de inicio: 15 de septiembre de 2024	**Fecha de fin:** 18 de septiembre de 2024
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Para permitir al usuario editar noticias existentes, se deben crear las rutas de acceso mediante el enrutador de Django. La vista debe recuperar los datos de la noticia de la base de datos, permitir la edición de los campos "Título", Çontenidoz Ïmagen", y actualizar la información en la base de datos. La plantilla a utilizar debe permitir cargar el formulario creado desde Python y validar los campos según los datos definidos.	

Tabla A.46. Tarea de desarrollo # 13

Tarea	
Número de tarea: 13	**Número de Historia de usuario:** 28
Nombre de la tarea: Visualizar Noticias	
Tipo de tarea: Funcionalidad	**Puntos estimados:** 0.2
Fecha de inicio: 21 de septiembre de 2024	**Fecha de fin:** 23 de septiembre de 2024
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Para permitir al usuario ver las noticias existentes, se deben crear las rutas de acceso mediante el enrutador de Django. La vista debe recuperar los datos de las noticias de la base de datos y mostrarlos en una interfaz gráfica.	

Tabla A.47. Tarea de desarrollo # 14

Tarea	
Número de tarea: 14	**Número de Historia de usuario:** 7
Nombre de la tarea: Monitorear estado de las bombas	
Tipo de tarea: Funcionalidad	**Puntos estimados:** 0.6
Fecha de inicio: 1 de septiembre de 2024	**Fecha de fin:** 21 de septiembre de 2024
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Para permitir al usuario ver el estado actual de las bombas de agua en tiempo real, se deben crear las rutas de acceso mediante el enrutador de Django. La vista debe recuperar y mostrar los datos del estado de las bombas en una interfaz gráfica.	

Tabla A.48. Tarea de desarrollo # 15

Tarea	
Número de tarea: 15	**Número de Historia de usuario:** 18
Nombre de la tarea: Eliminar afectación	
Tipo de tarea: Funcionalidad	**Puntos estimados:** 1
Fecha de inicio: 30 de septiembre de 2024	**Fecha de fin:** 13 de octubre de 2024
Programador responsable: Carlos Adrián Burley Ros	
Descripción: Para permitir al usuario eliminar afectaciones, se deben crear las rutas de acceso mediante el enrutador de Django. La vista debe proporcionar una interfaz gráfica que permita al usuario seleccionar y eliminar las afectaciones deseadas.	

Tabla A.49. Tarea de desarrollo # 16

Tarea	
Número de tarea: 16	**Número de Historia de usuario:** 2
Nombre de la tarea: Añadir usuario	
Tipo de tarea: Funcionalidad	**Puntos estimados:** 2
Fecha de inicio: 1 de septiembre de 2024	**Fecha de fin:** 14 de septiembre de 2024
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Para permitir al usuario crear nuevos usuarios, se deben crear las rutas de acceso mediante el enrutador de Django. La vista debe capturar los campos de usuario, rol y contraseña, y guardarlos en la base de datos. La plantilla a utilizar debe permitir cargar el formulario creado desde Python y validar los campos según los datos definidos.	

Tabla A.50. Tarea de desarrollo # 17

Tarea	
Número de tarea: 17	**Número de Historia de usuario:** 4

Continúa en la próxima página

Tabla A.50. Continuación de la página anterior

Nombre de la tarea: Eliminar usuario	
Tipo de tarea: Funcionalidad	**Puntos estimados:** 1
Fecha de inicio: 1 de septiembre de 2024	**Fecha de fin:** 14 de septiembre de 2024
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Para permitir al usuario, como administrador, eliminar usuarios del sistema, se deben crear las rutas de acceso mediante el enrutador de Django. La vista debe proporcionar una interfaz gráfica que permita al usuario seleccionar y eliminar los usuarios deseados.	

Tabla A.51. Tarea de desarrollo # 18

Tarea	
Número de tarea: 18	**Número de Historia de usuario:** 5
Nombre de la tarea: Monitoreo de presiones de agua	
Tipo de tarea: Funcionalidad	**Puntos estimados:** 3
Fecha de inicio: 1 de septiembre de 2024	**Fecha de fin:** 21 de septiembre de 2024
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Para permitir al usuario visualizar el registro en tiempo real de las presiones de agua en diferentes puntos del sistema, se deben crear las rutas de acceso mediante el enrutador de Django. La vista debe recuperar y mostrar los datos de presión en una interfaz gráfica.	

Tabla A.52. Tarea de desarrollo # 19

Tarea	
Número de tarea: 19	**Número de Historia de usuario:** 11
Nombre de la tarea: Realizar Alertas automáticas	
Tipo de tarea: Funcionalidad	**Puntos estimados:** 3
Fecha de inicio: 22 de septiembre de 2024	**Fecha de fin:** 13 de octubre de 2024
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Para permitir al usuario recibir alertas automáticas cuando se detecten anomalías en el funcionamiento del sistema de bombeo, se deben crear las rutas de acceso mediante el enrutador de Django. La vista debe proporcionar una interfaz gráfica que muestre las alertas según las condiciones críticas predefinidas.	

Tabla A.53. Tarea de desarrollo # 20

Tarea	
Número de tarea: 20	**Número de Historia de usuario:** 9
Nombre de la tarea: Filtrar automáticamente plantilla de control de operación diaria	
Tipo de tarea: Funcionalidad	**Puntos estimados:** 3

Continúa en la próxima página

Tabla A.53. Continuación de la página anterior

Fecha de inicio: 6 de octubre de 2024	**Fecha de fin:** 27 de octubre de 2024
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Para permitir el guardado automático de los datos de operación, se deben crear las rutas de acceso mediante el enrutador de Django. La vista debe capturar y guardar en la base de datos los datos como fecha, operadores, equipos en funcionamiento, presión de salida, nivel del tanque, frecuencia del variador y observaciones, sin requerir la intervención manual del usuario.	

Tabla A.54. Tarea de desarrollo # 21

Tarea	
Número de tarea: 21	**Número de Historia de usuario:** 29
Nombre de la tarea: Eliminar plantilla de control de operación	
Tipo de tarea: Funcionalidad	**Puntos estimados:** 1
Fecha de inicio: 22 de septiembre de 2024	**Fecha de fin:** 6 de octubre de 2024
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Para permitir al usuario eliminar las plantillas de control de operación, se deben crear las rutas de acceso mediante el enrutador de Django. La vista debe proporcionar una interfaz gráfica que permita al usuario seleccionar y eliminar las plantillas deseadas, mostrando una confirmación antes de realizar la eliminación.	

Tabla A.55. Tarea de desarrollo # 22

Tarea	
Número de tarea: 22	**Número de Historia de usuario:** 20
Nombre de la tarea: Recibir mensajes en tiempo real	
Tipo de tarea: Funcionalidad	**Puntos estimados:** 2
Fecha de inicio: 6 de octubre de 2024	**Fecha de fin:** 20 de octubre de 2024
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Para permitir a los usuarios (operadores y administradores) recibir y visualizar mensajes en tiempo real, se deben crear las rutas de acceso mediante el enrutador de Django. La vista debe proporcionar una interfaz gráfica que permita la recepción y visualización de los mensajes.	

Tabla A.56. Tarea de desarrollo # 23

Tarea	
Número de tarea: 23	**Número de Historia de usuario:** 21
Nombre de la tarea: Responder mensajes en tiempo real	

Continúa en la próxima página

Tabla A.56. Continuación de la página anterior

Tipo de tarea: Funcionalidad	**Puntos estimados:** 1
Fecha de inicio: 6 de octubre de 2024	**Fecha de fin:** 20 de octubre de 2024
Programador responsable: Carlos Adrián Burley Ross	
Descripción: Para permitir a los usuarios (operadores y administradores) responder y redactar mensajes en tiempo real, se deben crear las rutas de acceso mediante el enrutador de Django. La vista debe proporcionar una interfaz gráfica que permita la redacción y envío de mensajes.	

A.4. Pruebas de aceptación

Tabla A.57. Prueba de aceptación # 8

Caso de prueba de aceptación	
Código: HU8_P1	**Historia de usuario:** 8
Nombre: Monitoreo de Nivel del Tanque	
Descripción: Prueba para la funcionalidad: Monitoreo del Estado de Llenado del Tanque	
Condiciones de ejecución: • El usuario debe estar autenticado en el sistema. • Los sensores del tanque deben estar operativos.	
Pasos de ejecución: • El usuario accede a la sección de monitoreo. • El usuario selecciona la vista de estado del tanque. • El sistema muestra los datos en tiempo real.	
Resultados esperados: • El sistema muestra el nivel actual del tanque. • El sistema actualiza los datos automáticamente. • El sistema muestra alertas si los niveles son críticos. • El sistema registra los datos en el historial.	

Tabla A.58. Prueba de aceptación # 9

Caso de prueba de aceptación	
Código: HU 11_P 1	**Historia de usuario:** 11
Nombre: Exportación de Reporte de Afectaciones	
Descripción: Prueba para la funcionalidad: Descagar de Reportes de Afectaciones	
Condiciones de ejecución: • El usuario debe estar autenticado como Operador o Eléctrico. • Deben existir afectaciones registradas en el sistema.	

Continúa en la próxima página

Tabla A.58. Continuación de la página anterior

Pasos de ejecución: • El usuario accede a la sección de reportes. • El usuario selecciona el período de tiempo deseado. • El usuario selecciona el formato de exportación (PDF). • El usuario inicia la exportación.
Resultados esperados: • El sistema genera el reporte en el formato seleccionado. • El sistema incluye todas las afectaciones del período seleccionado. • El sistema permite la descarga del archivo generado. • El sistema registra la exportación en el historial.

Tabla A.59. Prueba de aceptación # 10

Caso de prueba de aceptación	
Código: HU 16_P 1	**Historia de usuario:** 16
Nombre: Crear Nuevo Usuario	
Descripción: Prueba para la funcionalidad: Añadir usuario	
Condiciones de ejecución: • El administrador debe estar autenticado en el sistema • El administrador debe tener permisos para gestionar usuarios	
Pasos de ejecución: • El administrador accede a la sección de gestión de usuarios • Selecciona la opción .Añadir nuevo usuario" • Completa el formulario con nombre de usuario, rol y contraseña • Confirma la creación del nuevo usuario	
Resultados esperados: • El sistema valida que la contraseña cumpla con los requisitos mínimos de seguridad • El sistema verifica que el nombre de usuario no exista previamente • El sistema crea el nuevo usuario y lo almacena en la base de datos • Se muestra un mensaje de confirmación exitosa	

Tabla A.60. Prueba de aceptación # 11

Caso de prueba de aceptación	
Código: HU 17_P 1	**Historia de usuario:** 17
Nombre: Modificación de Usuario	

Continúa en la próxima página

Tabla A.60. Continuación de la página anterior

Descripción: Prueba para la funcionalidad: Modificar usuario
Condiciones de ejecución: • El administrador debe estar autenticado en el sistema • Debe existir al menos un usuario para modificar • El administrador debe tener permisos de modificación
Pasos de ejecución: • El administrador accede a la lista de usuarios • Selecciona el usuario a modificar • Modifica los campos deseados (usuario, rol o contraseña) • Guarda los cambios
Resultados esperados: • El sistema valida los cambios realizados • Los datos del usuario se actualizan en la base de datos • Se muestra un mensaje de actualización exitosa • Los cambios se reflejan inmediatamente en la lista de usuarios

Tabla A.61. Prueba de aceptación # 12

Caso de prueba de aceptación	
Código: HU 18_P 1	**Historia de usuario:** 18
Nombre: Eliminación de Usuario	
Descripción: Prueba para la funcionalidad: Eliminar usuario	
Condiciones de ejecución: • El administrador debe estar autenticado en el sistema • Debe existir al menos un usuario para eliminar • El administrador debe tener permisos de eliminación	
Pasos de ejecución: • El administrador accede a la lista de usuarios • Selecciona el usuario a eliminar • Confirma la eliminación en el mensaje de advertencia	
Resultados esperados: • El sistema muestra un mensaje de confirmación antes de eliminar • El usuario se elimina correctamente de la base de datos • Se muestra un mensaje de eliminación exitosa • El usuario eliminado ya no aparece en la lista de usuarios	

Tabla A.62. Prueba de aceptación # 13

Caso de prueba de aceptación	
Código: HU 19_P 1	**Historia de usuario:** 19
Nombre: Monitoreo de Presiones	
Descripción: Prueba para la funcionalidad: Monitoreo de presiones de agua	
Condiciones de ejecución: • El usuario debe estar autenticado como Operador o Eléctrico • Los sensores de presión deben estar funcionando correctamente • Debe haber conexión con el sistema de monitoreo	
Pasos de ejecución: • El usuario accede a la sección de monitoreo de presiones • Visualiza los valores de presión en tiempo real • Verifica los diferentes puntos de medición	
Resultados esperados: • El sistema muestra las lecturas de presión actualizadas en tiempo real • Los valores se presentan en las unidades correctas • Se muestran claramente los diferentes puntos de medición • Las lecturas se actualizan automáticamente sin necesidad de refrescar la página	

Tabla A.63. Prueba de aceptación # 14

Caso de prueba de aceptación	
Código: HU 20_P 1	**Historia de usuario:** 20
Nombre: Sistema de Alertas	
Descripción: Prueba para la funcionalidad: Sistema de alertas automáticas	
Condiciones de ejecución: • El usuario debe estar autenticado como Operador o Eléctrico • Los sistemas de monitoreo deben estar activos • Deben estar configurados los umbrales de alerta	
Pasos de ejecución: • Se simula una condición crítica en el sistema • Se espera la generación automática de la alerta • El usuario recibe y visualiza la alerta	

Continúa en la próxima página

Tabla A.63. Continuación de la página anterior

Resultados esperados:
• El sistema detecta la condición crítica • Se genera una alerta automática • La alerta se muestra de forma visible en la interfaz • Se registra la alerta en el sistema • El usuario puede acknowledgar la recepción de la alerta

Tabla A.64. Prueba de aceptación # 15

Caso de prueba de aceptación	
Código: HU 21_P 1	**Historia de usuario:** 21
Nombre: Exportación de Planillas	
Descripción: Prueba para la funcionalidad: Descargar planilla de operaciones	
Condiciones de ejecución: • El usuario debe estar autenticado como Operador • Deben existir datos de operaciones para exportar • El usuario debe tener permisos de exportación	
Pasos de ejecución: • El usuario accede a la sección de planillas de operaciones • Selecciona el período de tiempo para la exportación • Selecciona el formato de exportación • Ejecuta la exportación	
Resultados esperados: • El sistema genera correctamente el archivo de exportación • El archivo contiene todos los datos del período seleccionado • Se descarga automáticamente el archivo • Se registra la actividad de exportación en el sistema	

Tabla A.65. Prueba de aceptación # 16

Caso de prueba de aceptación	
Código: HU 23_P 1	**Historia de usuario:** 23
Nombre: Guardado Automático de Plantilla	
Descripción: Prueba para la funcionalidad: Guardado automático de plantilla de control	

Continúa en la próxima página

Tabla A.65. Continuación de la página anterior

Condiciones de ejecución: • El sistema debe estar en funcionamiento • Debe haber espacio de almacenamiento disponible • La conexión a la base de datos debe estar activa
Pasos de ejecución: • El sistema recopila los datos de operación • Verifica la integridad de los datos • Ejecuta el guardado automático • Actualiza el registro de operaciones
Resultados esperados: • Los datos se guardan correctamente en la base de datos • Se mantiene un registro de la fecha y hora del guardado • Se confirma el éxito del guardado automático • Los datos guardados son accesibles para consulta posterior

Tabla A.66. Prueba de aceptación # 17

Caso de prueba de aceptación	
Código: HU 24_P 1	**Historia de usuario:** 24
Nombre: Eliminación de Plantilla	
Descripción: Prueba para la funcionalidad: Eliminación de plantilla de control	
Condiciones de ejecución: • El usuario debe estar autenticado como Operador • Debe existir al menos una plantilla para eliminar • El usuario debe tener permisos de eliminación	
Pasos de ejecución: • El usuario accede a la lista de plantillas • Selecciona la plantilla a eliminar • Confirma la eliminación en el diálogo de confirmación	
Resultados esperados: • El sistema muestra mensaje de confirmación antes de eliminar • La plantilla se elimina correctamente de la base de datos • Se muestra mensaje de eliminación exitosa • La plantilla ya no aparece en la lista	

Tabla A.67. Prueba de aceptación # 18

Caso de prueba de aceptación	
Código: HU 25_P 1	**Historia de usuario:** 25
Nombre: Recepción de Mensajes	
Descripción: Prueba para la funcionalidad: Recepción de mensajes en tiempo real	
Condiciones de ejecución: • El usuario debe estar autenticado como Operador o Administrador • Debe haber conexión activa al sistema de mensajería • El usuario debe tener permisos de mensajería	
Pasos de ejecución: • El usuario accede al sistema de mensajería • Se mantiene activa la conexión en tiempo real • Se recibe un nuevo mensaje	
Resultados esperados: • Los mensajes se reciben instantáneamente • Se muestra una notificación de nuevo mensaje • El mensaje se visualiza correctamente en la interfaz • Se actualiza el historial de mensajes	

Tabla A.68. Prueba de aceptación # 19

Caso de prueba de aceptación	
Código: HU 26_P 1	**Historia de usuario:** 26
Nombre: Respuesta de Mensajes	
Descripción: Prueba para la funcionalidad: Respuesta de mensajes en tiempo real	
Condiciones de ejecución: • El usuario debe estar autenticado en el sistema • Debe existir al menos un mensaje para responder • La conexión al sistema de mensajería debe estar activa	
Pasos de ejecución: • El usuario selecciona el mensaje a responder • Redacta la respuesta en el campo de texto • Envía la respuesta • Verifica la entrega del mensaje	

Continúa en la próxima página

Tabla A.68. Continuación de la página anterior

Resultados esperados: • El mensaje se envía instantáneamente • Se muestra confirmación de envío exitoso • El mensaje aparece en el historial de la conversación • El destinatario recibe la notificación en tiempo real

Tabla A.69. Prueba de aceptación # 20

Caso de prueba de aceptación	
Código: HU 27_P 1	**Historia de usuario:** 27
Nombre: Registro de Entrada	
Descripción: Prueba para la funcionalidad: Registro automático de entrada de operador	
Condiciones de ejecución: • El sistema debe estar funcionando correctamente • Debe estar configurado el horario establecido • El sistema de registro debe estar activo	
Pasos de ejecución: • El operador inicia sesión en el sistema • El sistema registra automáticamente la hora de entrada • Se verifica el horario contra el turno establecido	
Resultados esperados: • Se registra correctamente la hora de entrada • Se genera una alerta si la entrada está fuera del horario establecido • El registro queda almacenado en la base de datos • Se puede consultar el registro en el sistema	

Tabla A.70. Prueba de aceptación # 21

Caso de prueba de aceptación	
Código: HU 28_P 1	**Historia de usuario:** 28
Nombre: Registro de Entrada	
Descripción: Prueba para la funcionalidad: Registro automático de entrada de operador	
Condiciones de ejecución: • El sistema debe estar funcionando correctamente • Debe estar configurado el horario establecido • El sistema de registro debe estar activo	

Continúa en la próxima página

Tabla A.70. Continuación de la página anterior

Pasos de ejecución: • El operador inicia sesión en el sistema • El sistema registra automáticamente la hora de entrada • Se verifica el horario contra el turno establecido
Resultados esperados: • Se registra correctamente la hora de entrada • Se genera una alerta si la entrada está fuera del horario establecido • El registro queda almacenado en la base de datos • Se puede consultar el registro en el sistema

Tabla A.71. Prueba de aceptación # 22

Caso de prueba de aceptación	
Código: HU 29_P 1	**Historia de usuario:** 29
Nombre: Visualización de Registros de Acceso	
Descripción: Prueba para la funcionalidad: Visualización de registro de control de acceso	
Condiciones de ejecución: • El usuario debe estar autenticado como Administrador u Operador • Deben existir registros de entrada y salida • El usuario debe tener permisos para visualizar registros	
Pasos de ejecución: • El usuario accede a la sección de registros de control de acceso • Selecciona el período de tiempo a consultar • Visualiza el registro diario de entradas y salidas • Puede filtrar por operador o fecha si lo desea	
Resultados esperados: • Se muestran todos los registros del período seleccionado • La información se presenta de manera clara y organizada • Se pueden identificar fácilmente las entradas y salidas de cada operador • Los registros muestran la hora exacta de cada evento • Se pueden aplicar filtros para facilitar la búsqueda	

A.5. Pruebas unitarias

```
    user.save(using=self._db)
    user.save(using=self._db)
  File "D:\Programacion Web\1era practica\Mi tesis\GestionBombeo\gestion_agua\bombeo\models.py", line 32, in save
    raise ValidationError(f"El rol '{self.rol}' no es válido.")
django.core.exceptions.ValidationError: ["El rol '' no es válido."]

----------------------------------------------------------------------
Ran 8 tests in 27.166s

FAILED (errors=3)
Destroying test database for alias 'default'...
django.core.exceptions.ValidationError: ["El rol '' no es válido."]

----------------------------------------------------------------------
Ran 8 tests in 27.166s

FAILED (errors=3)
Destroying test database for alias 'default'...

----------------------------------------------------------------------
Ran 8 tests in 27.166s

FAILED (errors=3)
Destroying test database for alias 'default'...
----------------------------------------------------------------------
Ran 8 tests in 27.166s

FAILED (errors=3)
Destroying test database for alias 'default'...

FAILED (errors=3)
Destroying test database for alias 'default'...
```

Figura A.1. Código fuente de prueba unitarias

```
Found 5 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
F.FF.
======================================================================
FAIL: test_usuario_campo_nombre_longitud (bombeo.tests.test_name.UsuarioModelTest.test_usuario_campo_nombre_longitud)
Prueba que el campo nombre no exceda la longitud máxima.
----------------------------------------------------------------------
Traceback (most recent call last):
  File "D:\Programacion Web\1era practica\Mi tesis\GestionBombeo\gestion_agua\bombeo\tests\test_name.py", line 58, in test_usuario_campo_nombre_longitud
    with self.assertRaises(Exception):
AssertionError: Exception not raised

======================================================================
FAIL: test_usuario_creacion (bombeo.tests.test_name.UsuarioModelTest.test_usuario_creacion)
Prueba que un usuario puede ser creado correctamente.
----------------------------------------------------------------------
Traceback (most recent call last):
  File "D:\Programacion Web\1era practica\Mi tesis\GestionBombeo\gestion_agua\bombeo\tests\test_name.py", line 26, in test_usuario_creacion
    self.assertEqual(self.usuario.correo_electronico, self.usuario_data['correo_electronico'])
AssertionError: '' != 'testuser@example.com'
+ testuser@example.com


======================================================================
FAIL: test_usuario_roles_validos (bombeo.tests.test_name.UsuarioModelTest.test_usuario_roles_validos)
Prueba que los roles sean válidos.
----------------------------------------------------------------------
Traceback (most recent call last):
  File "D:\Programacion Web\1era practica\Mi tesis\GestionBombeo\gestion_agua\bombeo\tests\test_name.py", line 51, in test_usuario_roles_validos
    with self.assertRaises(ValueError):
AssertionError: ValueError not raised

----------------------------------------------------------------------
Ran 5 tests in 16.369s

FAILED (failures=3)
```

Figura A.2. Código fuente de prueba unitarias

```python
    def test_usuario_correo_unico(self):
        """Prueba que el campo correo electrónico sea único."""
        with self.assertRaises(Exception):
            Usuario.objects.create_user(
                username='testuser2',
                email=self.usuario_data['correo_electronico'],  # Mismo correo
                password='securepassword2',
                nombre='Test User 2',
                rol='usuario'
            )

    def test_usuario_roles_validos(self):
        """Prueba que los roles sean válidos."""
        self.usuario.rol = 'tecnico'
        self.usuario.save()
        self.assertEqual(self.usuario.rol, 'tecnico')

        with self.assertRaises(ValueError):
            self.usuario.rol = 'invalid_role'
            self.usuario.save()

    def test_usuario_campo_nombre_longitud(self):
        """Prueba que el campo nombre no exceda la longitud máxima."""
        self.usuario.nombre = 'a' * 101  # Excede el límite de 100 caracteres
        with self.assertRaises(Exception):
            self.usuario.save()
```

Figura A.3. Código fuente de prueba unitarias

```python
from django.test import TestCase
from bombeo.models import Usuario

class UsuarioModelTest(TestCase):

    def setUp(self):
        """Configura datos iniciales para las pruebas."""
        self.usuario_data = {
            'username': 'testuser',
            'nombre': 'Test User',
            'correo_electronico': 'testuser@example.com',
            'rol': 'administrador'
        }
        self.usuario = Usuario.objects.create_user(
            username=self.usuario_data['username'],
            email=self.usuario_data['correo_electronico'],
            password='securepassword',
            nombre=self.usuario_data['nombre'],
            rol=self.usuario_data['rol']
        )

    def test_usuario_creacion(self):
        """Prueba que un usuario puede ser creado correctamente."""
        self.assertEqual(self.usuario.username, self.usuario_data['username'])
        self.assertEqual(self.usuario.nombre, self.usuario_data['nombre'])
        self.assertEqual(self.usuario.correo_electronico, self.usuario_data['correo_electronico'])
        self.assertEqual(self.usuario.rol, self.usuario_data['rol'])
        self.assertTrue(self.usuario.check_password('securepassword'))

    def test_usuario_str(self):
        """Prueba el método __str__."""
        self.assertEqual(str(self.usuario), self.usuario_data['username'])
```

Figura A.4. Código fuente de prueba unitarias

```
(env) D:\Programacion Web\1era practica\Mi tesis\GestionBombeo\gestion_agua>py manage.py test
Found 10 test(s).
(env) D:\Programacion Web\1era practica\Mi tesis\GestionBombeo\gestion_agua>py manage.py test
Found 10 test(s).
Found 10 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..........
----------------------------------------------------------------------
Ran 10 tests in 27.797s

OK
System check identified no issues (0 silenced).
..........
----------------------------------------------------------------------
Ran 10 tests in 27.797s

OK
..........
```

Figura A.5. Código fuente de prueba unitarias

```python
from django.test import TestCase
from django.utils import timezone
from bombeo.models import Variador, Bomba

class VariadorTestCase(TestCase):
    def setUp(self):
        # Crear algunas bombas para las pruebas
        Bomba.objects.create(nombre="Bomba 1", estado=False)
        Bomba.objects.create(nombre="Bomba 2", estado=False)
        Bomba.objects.create(nombre="Bomba 3", estado=False)

        # Crear una instancia de Variador para las pruebas
        self.variador = Variador.objects.create(
            presion_actual=0.00,
            frecuencia=50.0,
            timestamp=timezone.now()
        )

    def test_variador_str(self):
        # Verificar la representación en cadena del objeto
        expected_str = f"Presión de Salida: {self.variador.presion_actual} PSI, Tiempo: {self.variador.timestamp}"
        self.assertEqual(str(self.variador), expected_str)

    def test_ajustar_presion(self):
        # Ajustar la presión y verificar que se actualice correctamente
        self.variador.ajustar_presion(30)
        self.assertEqual(self.variador.presion_actual, 30.00)

        # Verificar el estado de las bombas
        bombas = Bomba.objects.all()
        self.assertTrue(bombas[0].estado)  # Bomba 1 debe estar encendida
        self.assertTrue(bombas[1].estado)  # Bomba 2 debe estar encendida
```

Figura A.6. Código fuente de prueba unitarias

```python
    def test_ajustar_presion(self):
        # Ajustar la presión y verificar que se actualice correctamente
        self.variador.ajustar_presion(30)
        self.assertEqual(self.variador.presion_actual, 30.00)

        # Verificar el estado de las bombas
        bombas = Bomba.objects.all()
        self.assertTrue(bombas[0].estado)  # Bomba 1 debe estar encendida
        self.assertTrue(bombas[1].estado)  # Bomba 2 debe estar encendida
        self.assertTrue(bombas[2].estado)  # Bomba 3 debe estar encendida

    def test_ajustar_presion_con_bombas_apagadas(self):
        # Ajustar la presión a un valor bajo y verificar que se enciendan las bombas correctas
        self.variador.ajustar_presion(10)

        # Verificar el estado de las bombas
        bombas = Bomba.objects.all()
        self.assertTrue(bombas[0].estado)  # Bomba 1 debe estar encendida
        self.assertFalse(bombas[1].estado)  # Bomba 2 debe estar apagada
        self.assertFalse(bombas[2].estado)  # Bomba 3 debe estar apagada
```

Figura A.7. Código fuente de prueba unitarias

```
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..............presion actual: 30
Bombas encendidas: 3
<QuerySet [<Bomba: Bomba: Bomba 1, Apagada>, <Bomba: Bomba: Bomba 2, Apagada>, <Bomba: Bomba: Bomba 3, Apagada>]>
.presion actual: 10
Bombas encendidas: 1
<QuerySet [<Bomba: Bomba: Bomba 1, Apagada>, <Bomba: Bomba: Bomba 2, Apagada>, <Bomba: Bomba: Bomba 3, Apagada>]>
..
----------------------------------------------------------------------
Ran 18 tests in 27.396s

OK
Destroying test database for alias 'default'...

(env) D:\Programacion Web\1era practica\Mi tesis\GestionBombeo\gestion_agua>
```

Figura A.8. Código fuente de prueba unitarias

```python
from django.test import TestCase
from django.utils import timezone
from bombeo.models import Noticia

class NoticiaTestCase(TestCase):
    def setUp(self):
        # Crear un registro inicial de Noticia
        self.noticia = Noticia.objects.create(
            titulo='Noticia de Prueba',
            contenido='Este es el contenido de la noticia de prueba.',
            fecha_publicacion=timezone.now()
        )

    def test_noticia_str(self):
        # Verificar la representación en cadena del objeto
        self.assertEqual(str(self.noticia), self.noticia.titulo)

    def test_noticia_creacion(self):
        # Verificar que el objeto se haya creado correctamente
        self.assertEqual(self.noticia.titulo, 'Noticia de Prueba')
        self.assertEqual(self.noticia.contenido, 'Este es el contenido de la noticia de prueba.')
        self.assertIsInstance(self.noticia.fecha_publicacion, timezone.datetime)

    def test_noticia_imagen_opcional(self):
        # Verificar que la imagen opcional esté en blanco
        self.assertFalse(self.noticia.imagen)  # Esto será True si no hay imagen
```

Figura A.9. Código fuente de prueba unitarias

```python
from django.test import TestCase
from django.utils import timezone
from bombeo.models import RegistroOperacional

class RegistroOperacionalTestCase(TestCase):
    def setUp(self):
        # Crear un registro inicial de RegistroOperacional
        self.registro = RegistroOperacional.objects.create(
            hora=timezone.now(),
            presion_entrada=50.00,
            presion_salida=45.00,
            nivel_tanque=75.00,
            frecuencia_variador=60.00,
            volumen_bombeado=1000.00,
            bombas_trabajando=2
        )

    def test_registro_str(self):
        # Verificar la representación en cadena del objeto
        expected_str = f"Registro {self.registro.hora.strftime('%Y-%m-%d %H:%M:%S')}: Presión Entrada {self.registro.pr
        self.assertEqual(str(self.registro), expected_str)

    def test_registro_creacion(self):
        # Verificar que el objeto se haya creado correctamente
        self.assertEqual(self.registro.presion_entrada, 50.00)
        self.assertEqual(self.registro.presion_salida, 45.00)
        self.assertEqual(self.registro.nivel_tanque, 75.00)
        self.assertEqual(self.registro.frecuencia_variador, 60.00)
        self.assertEqual(self.registro.volumen_bombeado, 1000.00)
        self.assertEqual(self.registro.bombas_trabajando, 2)
```

Figura A.10. Código fuente de prueba unitarias

```python
from django.test import TestCase
from django.utils import timezone
from bombeo.models import VolumenBombeado

class VolumenBombeadoTestCase(TestCase):
    def setUp(self):
        # Crear un registro inicial de VolumenBombeado
        self.volumen_bombeado = VolumenBombeado.objects.create(
            volumen_actual=100.00,
            timestamp=timezone.now()
        )

    def test_volumen_bombeado_str(self):
        # Verificar la representación en cadena del objeto
        expected_str = f"VolumenBombeo {self.volumen_bombeado.volumen_actual} - {self.volumen_bombeado.timestamp}"
        self.assertEqual(str(self.volumen_bombeado), expected_str)

    def test_volumen_bombeado_creacion(self):
        # Verificar que el objeto se haya creado correctamente
        self.assertEqual(self.volumen_bombeado.volumen_actual, 100.00)
        self.assertIsInstance(self.volumen_bombeado.timestamp, timezone.datetime)

    def test_volumen_bombeado_actualizacion(self):
        # Actualizar el volumen y verificar que se guarda correctamente
        self.volumen_bombeado.volumen_actual = 150.00
        self.volumen_bombeado.save()
        self.assertEqual(self.volumen_bombeado.volumen_actual, 150.00)
```

Figura A.11. Código fuente de prueba unitarias

```
----------------------------------------------------------------------
Ran 23 tests in 27.667s

OK
Destroying test database for alias 'default'...

(env) D:\Programacion Web\1era practica\Mi tesis\GestionBombeo\gestion_agua>py manage.py test
Found 26 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.....................presion actual: 30
Bombas encendidas: 3
<QuerySet [<Bomba: Bomba: Bomba 1, Apagada>, <Bomba: Bomba: Bomba 2, Apagada>, <Bomba: Bomba: Bomba 3, Apagada>]>
.presion actual: 10
Bombas encendidas: 1
<QuerySet [<Bomba: Bomba: Bomba 1, Apagada>, <Bomba: Bomba: Bomba 2, Apagada>, <Bomba: Bomba: Bomba 3, Apagada>]>
.....
----------------------------------------------------------------------
Ran 26 tests in 27.985s

OK
Destroying test database for alias 'default'...
```

Figura A.12. Código fuente de prueba unitarias

```python
from django.test import TestCase
from bombeo.models import Bomba

class BombaTestCase(TestCase):
    def setUp(self):
        # Crear una bomba para las pruebas
        self.bomba = Bomba.objects.create(
            nombre="Bomba Principal",
            estado=True
        )

    def test_bomba_str(self):
        # Verificar la representación en cadena del objeto
        expected_str = "Bomba: Bomba Principal, Encendida"
        self.assertEqual(str(self.bomba), expected_str)

    def test_bomba_estado_apagada(self):
        # Crear una bomba apagada y verificar su representación
        bomba_apagada = Bomba.objects.create(
            nombre="Bomba Secundaria",
            estado=False
        )
        expected_str = "Bomba: Bomba Secundaria, Apagada"
        self.assertEqual(str(bomba_apagada), expected_str)

    def test_bomba_creacion(self):
        # Verificar que la bomba se ha creado correctamente
        self.assertEqual(self.bomba.nombre, "Bomba Principal")
        self.assertTrue(self.bomba.estado)
```

Figura A.13. Código fuente de prueba unitarias

```
(env) D:\Programacion Web\1era practica\Mi tesis\GestionBombeo\gestion_agua>py manage.py test
Found 13 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.............
----------------------------------------------------------------------
Ran 13 tests in 27.236s

OK
Destroying test database for alias 'default'...

(env) D:\Programacion Web\1era practica\Mi tesis\GestionBombeo\gestion_agua>
```

Figura A.14. Código fuente de prueba unitarias

```python
from django.test import TestCase
from bombeo.models import HorarioBombeo, crear_horarios_por_defecto
import datetime

class HorarioBombeoTestCase(TestCase):
    def setUp(self):
        # Crear un horario de bombeo para las pruebas usando datetime.time
        self.horario = HorarioBombeo.objects.create(
            hora_inicio=datetime.time(6, 0),  # 06:00
            hora_fin=datetime.time(10, 0),    # 10:00
            presion=30
        )

    def test_horario_bombeo_str(self):
        # Verificar la representación en cadena del objeto
        expected_str = "Horario: 06:00 - 10:00, Presión: 30 kPa"
        self.assertEqual(str(self.horario), expected_str)

    def test_crear_horarios_por_defecto(self):
        # Verificar que la función devuelve los horarios por defecto
        horarios = crear_horarios_por_defecto()
        self.assertEqual(len(horarios), 6)  # Verificar que hay 6 horarios
        self.assertEqual(horarios[0], {"hora_inicio": "06:00", "hora_fin": "10:00", "presion": 30})
        self.assertEqual(horarios[1], {"hora_inicio": "10:00", "hora_fin": "12:00", "presion": 10})
```

Figura A.15. Código fuente de prueba unitarias

```
(env) D:\Programacion Web\1era practica\Mi tesis\GestionBombeo\gestion_agua>py manage.py test
Found 10 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.F........
======================================================================
FAIL: test_horario_bombeo_str (bombeo.tests.test_horario.HorarioBombeoTestCase.test_horario_bombeo_str)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "D:\Programacion Web\1era practica\Mi tesis\GestionBombeo\gestion_agua\bombeo\tests\test_horario.py", line 16, in test_horario_bombeo_str
    self.assertEqual(str(self.horario), expected_str)
AssertionError: 'Horario: 06:00 - 10:00, Presión: 30 kPa' != 'Horario: 06:00:00 - 10:00:00, Presión: 30 kPa'
- Horario: 06:00 - 10:00, Presión: 30 kPa
+ Horario: 06:00:00 - 10:00:00, Presión: 30 kPa
?             +++       +++


----------------------------------------------------------------------
Ran 10 tests in 29.273s

FAILED (failures=1)
Destroying test database for alias 'default'...

(env) D:\Programacion Web\1era practica\Mi tesis\GestionBombeo\gestion_agua>
```

Figura A.16. Código fuente de prueba unitarias

```python
    def test_noticia_imagen_opcional(self):
        # Verificar que la imagen opcional esté en blanco
        self.assertIsNone(self.noticia.imagen)

    def test_noticia_fecha_publicacion_default(self):
        # Crear una noticia sin especificar la fecha de publicación
        noticia_sin_fecha = Noticia.objects.create(
            titulo='Otra Noticia',
            contenido='Contenido de otra noticia.'
        )
        self.assertIsNotNone(noticia_sin_fecha.fecha_publicacion)
```

Figura A.17. Código fuente de prueba unitarias

```python
from django.test import TestCase
from django.utils import timezone
from bombeo.models import Noticia

class NoticiaTestCase(TestCase):
    def setUp(self):
        # Crear un registro inicial de Noticia
        self.noticia = Noticia.objects.create(
            titulo='Noticia de Prueba',
            contenido='Este es el contenido de la noticia de prueba.',
            fecha_publicacion=timezone.now()
        )

    def test_noticia_str(self):
        # Verificar la representación en cadena del objeto
        self.assertEqual(str(self.noticia), self.noticia.titulo)

    def test_noticia_creacion(self):
        # Verificar que el objeto se haya creado correctamente
        self.assertEqual(self.noticia.titulo, 'Noticia de Prueba')
        self.assertEqual(self.noticia.contenido, 'Este es el contenido de la noticia de prueba.')
        self.assertIsInstance(self.noticia.fecha_publicacion, timezone.datetime)

    def test_noticia_imagen_opcional(self):
        # Verificar que la imagen opcional esté en blanco
        self.assertIsNone(self.noticia.imagen)
```

Figura A.18. Código fuente de prueba unitarias