UNIVERSIDAD NACIONAL DE SAN CRISTÓBAL DE HUAMANGA

FACULTAD DE INGENIERÍA DE MINAS, GEOLOGÍA Y CIVIL

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS:

Bot en telegram para registro de worklogs en Jira en GDN Perú NTT Data, 2024

Para optar el título profesional de:
INGENIERO DE SISTEMAS

PRESENTADO POR:
Bach. Julio Cesar CISNEROS PALOMINO

ASESOR:
Mtra. Elinar CARRILLO RIVEROS

AYACUCHO - PERÚ

2025

Dedicatoria

A mi familia por las muestras de afecto y apoyo incondicional en todo mi proceso educativo.

A todos mis amigos y allegados que influyeron desinteresadamente en la realización de este proyecto.

Agradecimiento

La realización de este trabajo no hubiera sido posible sin el apoyo moral e incondicional de mi familia.

Agradezco a la Universidad Nacional de San Cristóbal de Huamanga por los años de enseñanza profesional y crecimiento personal.

A la plana docente de la escuela profesional de Ingeniería de Sistemas, quienes, con su enseñanza continua, conocimiento y experiencia a lo largo de toda la etapa universitaria estuvieron guiándome.

Resumen

La empresa GDN NTTDATA PERU es un innovador Global Delivery Network confiable de servicios de tecnología de la información y negocios con sede en Trujillo y Arequipa. Ayudan a los clientes a transformarse a través de consultoría, soluciones industriales, servicios de procesos comerciales, modernización digital e informática y servicios gestionados. De acuerdo con su procedimiento indica que todo colaborador debe de registrar su avance de trabajo diario en la herramienta de Jira Kosin, esto se realiza por un solo medio que es el Jira Web y solo se puede realizar desde la PC de la empresa, esto genera que cada colaborador que no tenga acceso a su equipo no cumpla con el registro de sus actividades diarias y salen en los reportes de registro que generan el área de calidad interna.

El estudio tiene como propósito desarrollar Bot en telegram para el registro de los worklogs, utilizando un lenguaje de programación orientado a objetos en este caso JavaScript, base de datos no relacional como es MongoDB, IDE de desarrollo que es Visual Studio Code, Api de Telegram, Api de Jira y las aplicaciones de Microsoft Office 365(Teams, Power Automted y Outlook).

La presente investigación se realizará solo para el registro de worklogs en jira GDN PERU, se utilizará la metodología de desarrollo de software XP. El tipo de Investigación es: observacional, prospectivo y descriptivo.

Los principales resultados serán; disminuir el tiempo de registro de worklogs en jira, Generar reportes de registro de worklogs en jira.

Palabras Claves: Registro worklogs, tiempo de registro, reporte de registro.

Abstract

GDN NTTDATA PERU is an innovative and reliable Global Delivery Network for information technology and business services, with headquarters in Trujillo and Arequipa. The company helps clients transform through consulting, industry-specific solutions, business process services, digital and IT modernization, and managed services. According to its internal procedures, every employee is required to record their daily work progress using the Jira Kosin tool. This can only be done through the Jira Web platform and exclusively from the company-issued PC. As a result, employees who do not have access to their device are unable to complete their daily activity logs, which is reflected in the registration reports generated by the internal quality department.

The purpose of this study is to develop a Telegram bot for the registration of worklogs, using an object-oriented programming language—in this case, JavaScript—a non-relational database such as MongoDB, the Visual Studio Code development IDE, the Telegram API, the Jira API, and Microsoft Office 365 applications (Teams, Power Automate, and Outlook).

This research will focus exclusively on the registration of worklogs in Jira GDN PERU, using the XP (Extreme Programming) software development methodology. The type of research is observational, prospective, and descriptive.

The main expected outcomes are: reducing the time required to register worklogs in Jira, and generating reports of worklog entries in Jira.

Keywords: Worklog registration, registration time, registration report.

Índice

Lista de Tablas

Lista de Figuras

CAPÍTULO I
INTRODUCCIÓN

39.1 PLANTEAMIENTO DEL PROBLEMA

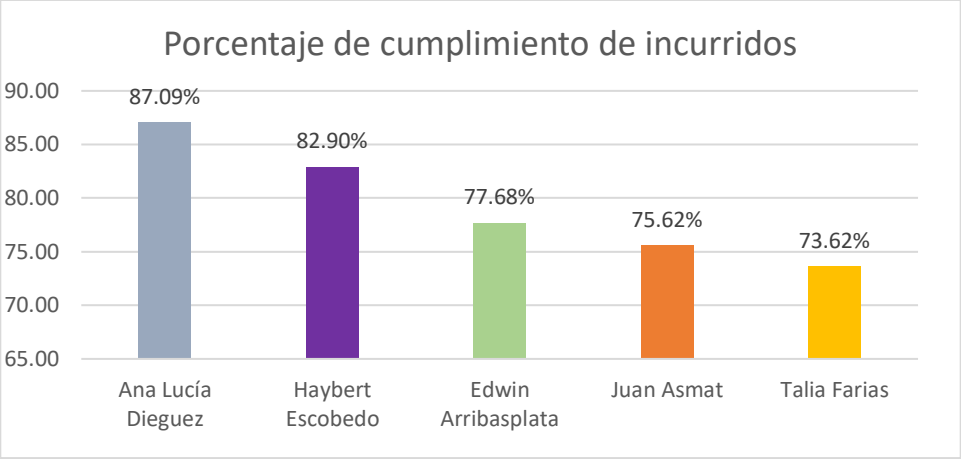
La empresa GDN NTTDATA PERU es un innovador Global Delivery Network confiable de servicios de tecnología de la informacion y negocios con sede en Trujillo y Arequipa. Ayudan a los clientes a transformarse a través de consultoría, soluciones industriales, servicios de procesos comerciales, modernización digital e informática y servicios gestionados empresa, es una de las empresas reconocidas con la certificación Global de Top Employer otorgada por el Top Employers Institute, consolidándose como uno de los mejores lugares para trabajar.

Según el procedimiento de la empresa es importante mencionar que todo colaborador de GDN PERU debe registrar su avance de trabajo diario en la herramienta de Jira Kosin, para poder transparentar y demostrar el trabajo con la finalidad de poder ver los avances realizados en la tarea o proyecto asignado, en este sentido los responsables de cada proyecto analizaran y determinaran grado de eficiencia e ineficiencia, tambien podrán coordinar y gestionar acciones preventivas en la calidad del servicio.

El registro de los worklogs en el mes de diciembre del 2023 solo alcanzo el 79.44%, lo cual hay un 20.56% que no se ha registrado debidamente en el jira kosin, esto afecta negativamente en los proyectos ya que no se podrá determinar el grado de eficiencia de cada tarea asignada a un colaborador.

Figura 1

Gráfico de barras de porcentaje de cumplimiento de incurridos, DIC2023



Fuente: Calidad Interna nttdata – Jira Kosin, (2023).

Tabla 1. Incurrido planificado por managers de gdn nttdata, DIC2023

Colaboradores	Incurrido (Real)	Incurrido (Planificado)	Porcentaje (%)
Ana Lucía Dieguez	35,270	40,500	87.09%
Haybert Escobedo	28,629	34,533	82.90%
Edwin Arribasplata	25,650	33,021	77.68%
Juan Asmat	27,154	35,908	75.62%
Talia Farias	29,698	40,338	73.62%
Total, general	146,401	184,300	79.44%

Fuente: Calidad Interna nttdata – Jira Kosin, (2023).

Es por ello que, que es necesario implementar un bot que sea capaz de hacer el registro de worklogs para mejorar la calidad del dato en cuestión a las horas reales y planificadas, así mismo poder sacar reportes de lo registrado y alertar a los que no lo van haciendo.

39.2 FORMULACIÓN DEL PROBLEMA

39.2.1 Problema general

¿De qué manera realizar registro de worklogs en jira de forma más eficiente en GDN

PERU NTTDATA, 2024?

39.2.2 Problemas específicos

a. ¿De qué manera gestionar el tiempo de registro de worklogs en jira?
b. ¿Cómo realizar reportes de registro de worklogs en jira?

1.1. JUSTIFICACIÓN DE LA INVESTIGACIÓN

La implementación de un bot en telegram beneficiara a los colaboradores de gdn nttdata Perú en poder incurrir sus worklog de manera accesible y más rápida. También a los supervisores de cada proyecto donde podrán ver incurridos de sus colaboradores a cargo y también poder enviar alertas tempranas para que puedan incurrir a tiempo.

Por otro lado, también podrán realizar reportes de sus incurridos y observar las tareas que tengan asignados en cada proyecto, esto les permitirá tener la información a la mano y no va a ser necesario que ingresen a la laptop corporativa para que puedan registrar y/o sacar un reporte de sus worklogs.

39.3 LIMITACIONES DE LA INVESTIGACIÓN

La presente investigación se considera solo en gdn nttdata Perú que consta de la ciudad de Lima y Arequipa, sin embargo, debería estudiarse en todos los gdn a nivel mundial que son Alicante, Brasil, Chile, México, Moroco, Murcia, Portugal, Salamanca y Sevilla, entonces los resultados tendrán un margen de error que debe ser controlado durante el proceso de investigación.

39.4 OBJETIVOS

39.4.1 Objetivo general

Desarrollar un bot en telegram para realizar registro de worklogs en jira de forma más eficiente en GDN PERU NTTDATA, 2024.

39.4.2 Objetivo específico

a. Gestionar el tiempo de registro de worklogs en jira
b. Generar reportes de registro de worklogs en jira

CAPÍTULO II

MARCO TEÓRICO

40.1 ANTECEDENTES DE LA INVESTIGACIÓN

Según David (2009) define que el registro de trabajo debe alinearse estrechamente con los objetivos estratégicos de la organización y que los empleados deben ser responsables de contribuir directamente a esos objetivos a través de sus funciones y responsabilidades. De igual manera, Drucker (1974), aclara que el registro de trabajos es esencial para comprender y optimizar la eficiencia organizativa, también enfatiza la importancia de definir claramente las responsabilidades y las tareas asociadas con cada puesto de trabajo, así como de establecer métricas claras para evaluar el rendimiento. Asimismo, Taylor (1911), enfatiza que el enfoque del registro de trabajo se centra en la descripción detallada de cada tarea y en la identificación de los métodos más eficientes para realizarlas.

Según Allen (2001), enfatiza la importancia de registrar el tiempo dedicado a cada tarea como parte del proceso de organización personal, sugiere que mantener un registro detallado de las actividades ayuda a identificar áreas de mejora en la gestión del tiempo y priorizar las tareas de manera más efectiva. Por su parte Vanderkam (2010), argumenta que todos tenemos las mismas 168 horas a la semana y que como las utilizamos determina nuestra productividad y satisfacción personal, sugiere llevar un registro detallado del tiempo para identificar patrones y oportunidades de optimización.

Allen (2001), menciona un reporte de registro detallado de las actividades ayuda a mantenerse al tanto de las responsabilidades y a tomar decisiones informadas sobre como asignar el tiempo y los recursos. A todo esto, Covey (1989), menciona que un reporte de registro de actividades puede ayudar a evaluar si las tareas realizadas están alineadas con los objetivos personales y profesionales establecidos. Conviene precisar

que Vanderkam (2010), comenta que un reporte de registro de actividades puede ayudar a identificar oportunidades para reasignar el tiempo y maximizar la productividad.

40.2 MARCO CONCEPTUAL

40.2.1 Registro de Worklogs

Según David (2009) define que el registro de trabajo debe alinearse estrechamente con los objetivos estratégicos de la organización y que los empleados deben ser responsables de contribuir directamente a esos objetivos a través de sus funciones y responsabilidades. De igual manera, Drucker (1974), aclara que el registro de trabajos es esencial para comprender y optimizar la eficiencia organizativa, también enfatiza la importancia de definir claramente las responsabilidades y las tareas asociadas con cada puesto de trabajo, asi como de establecer metricas claras para evaluar el rendimiento. A si mismo, Taylor (1911), enfatiza que el enfoque del registro de trabajo se centra en la descripción detallada de cada tarea y en la identificación de los métodos más eficientes para realizarlas.

40.2.2 Tiempo de registro

Según Allen (2001), enfatiza la importancia de registrar el tiempo dedicado a cada tarea como parte del proceso de organización personal, sugiere que mantener un registro detallado de las actividades ayuda a identificar áreas de mejora en la gestión del tiempo y priorizar las tareas de manera más efectiva. Por su parte Vanderkam (2010), argumenta que todos tenemos las mismas 168 horas a la semana y que como las utilizamos determina nuestra productividad y satisfacción personal, sugiere llevar un registro detallado del tiempo para identificar patrones y oportunidades de optimización.

40.2.3 Reporte de registro

Allen (2001), menciona un reporte de registro detallado de las actividades ayuda a mantenerse al tanto de las responsabilidades y a tomar decisiones informadas sobre como asignar el tiempo y los recursos. A todo esto, Covey (1989), menciona que un reporte de registro de actividades puede ayudar a evaluar si las tareas realizadas estan alineadas con los objetivos personales y profesionales establecidos. Conviene precisar que Vanderkam (2010), comenta que un reporte de registro de actividades puede ayudar a identificar oportunidades para reasignar el tiempo y maximizar la productividad

40.3 MARCO REFERENCIAL

40.3.1 Bot Telegram

Telegram Bot API (mencionado también como Telegram Bot Platform) es el nombre oficial de la interfaz de programación del servicio de mensajería Telegram orientada a los bots (abreviación de robot). La interfaz está operativa para varios clientes de la

aplicación a través de un nombre de usuario específico. El mecanismo, que forma parte de MTProto, hace hincapié en la creación de cuentas bot, usuarios autónomos que se controlan por varias acciones: botones interactivos, secuencias o por inteligencia artificial. Están disponible en todos los clientes a través de un alias terminado en bot. También se destaca por su libertad en el desarrollo al ser multilenguaje. Debido a su proliferación mayoritaria de terceros para todo tipo de uso, los medios catalogan a Telegram como pionero en la masificación de bots conversacionales. (Wikipedia s.f., 2024).

40.3.2 Jira

Jira es un producto de software propietario para la gestión de proyectos, seguimiento de errores e incidencias. La herramienta fue desarrollada por la empresa australiana Atlassian. Inicialmente se utilizó para el desarrollo de software, sirviendo de apoyo para la gestión de requisitos, seguimiento del estado de desarrollo y más tarde para la gestión de errores. Jira puede ser utilizado para la gestión y mejora de los procesos, gracias a sus funciones para la organización de flujos de trabajo. (Wikipedia s.f., 2024).

40.3.3 Tecnologías de Internet

Internet, también llamado autopista de información, designa un conjunto de redes informáticas relacionadas entre sí, para permitir que los usuarios puedan comunicarse entre sí; es una red abierta. Su principio básico es la trasmisión de datos de manera fiable entre ordenadores (Colección Esencial, 2011). A demas Luján (2001), asevera que contrario a otro servicio online, que se controlan de forma centralizada, la Internet posee un diseño descentralizado. Dado que cada ordenador (host) en la Internet es independiente. Los operadores pueden elegir qué servicio usar y qué servicios locales proporcionar. De igual forma Cafassi (1998) asevera que Internet es una tecnología. Sin embargo, no es la acepción más correcta y su consecuencia se ven a la hora de analizar las implicaciones sociales que conlleva su uso.

40.3.4 Lenguaje de Programación Orientado a Objetos

Weitzenfeld (s.f.), asevera que los lenguajes de programación orientados a objetos varían en sus estructuras y flujos de control. Y a pesar de que estos estén diseñados para un mismo tipo de programación, existen aspectos que hacen que ciertos lenguajes ofrezcan mejor apoyo que otros durante el desarrollo de un sistema de softwarea.

“Es un conjunto de símbolos, palabras y reglas que permiten implementar un algoritmo en una computadora. Dicha implementación se conoce como programa y se escribe como una secuencia de frases del lenguaje de programación” (Osorio, s.f., p.368).

Los lenguajes de programación orientados a objetos ofrecen las ventajas potenciales de los códigos reutilizables, costos inferiores, menos pruebas y rapidez en la puesta en marcha. Los programadores pueden combinar, modificar e integrar módulos pre-desarrollados en un programa unificado. Entro los más importantes están: Smalltalk, C++ y Java. (Stair y Reynolds, 1999, p.461).

Para Craing (2002), los lenguajes orientados a objetos están definidos por un conjunto de propiedades. La medida en la que un lenguaje particular satisfaga estas propiedades define en cuanto es un lenguaje orientado a objetos.

40.3.5 Lenguaje de Programación Lógica

La programación lógica, junto con la funcional, forma parte de lo que se conoce como programación declarativa. En los lenguajes tradicionales, la programación consiste en indicar cómo resolver un problema mediante sentencias; en la programación lógica, se trabaja de una forma descriptiva, estableciendo relaciones entre entidades, indicando no cómo, sino qué hacer (Bratko, 2002).

Programación lógica. Consiste en la aplicación del corpus de conocimiento sobre lógica para el diseño de lenguajes de programación; no debe confundirse con la disciplina de la lógica computacional. La programación lógica es un tipo de paradigmas de programación dentro del paradigma de programación declarativa (Sipser, 1996).

40.3.6 Base de Datos no Relacional

Según Redmond y Wilson (2012), una base de datos no relacional, también conocida como base de datos NoSQL, es un sistema de gestión de datos que difiere del enfoque tradicional de las bases de datos relacionales al no utilizar un modelo basado en tablas con relaciones predefinidas. En cambio, las bases de datos no relacionales adoptan una variedad de modelos de datos flexibles y escalables que pueden adaptarse mejor a las necesidades de diferentes tipos de aplicaciones y conjuntos de datos. Estas bases de datos NoSQL pueden basarse en modelos de datos de documentos, grafos, clave-valor o columnas, permitiendo a los desarrolladores almacenar y manipular datos de manera eficiente en entornos distribuidos y de alta disponibilidad.

40.3.7 Mongodb

Según Chodorow y Dirolf (2013), MongoDB es un sistema de gestión de bases de datos NoSQL, diseñado para almacenar datos de forma flexible y escalable, utilizando un modelo de documentos JSON (JavaScript Object Notation). En MongoDB, los datos se

organizan en colecciones, que a su vez contienen documentos individuales. Cada documento puede tener un esquema flexible y puede contener diferentes tipos de datos, lo que permite una fácil adaptación a las necesidades cambiantes de las aplicaciones. Destaca por su capacidad de escalamiento horizontal, su alto rendimiento y su capacidad de manejar grandes volúmenes de datos no estructurados en entornos distribuidos.

40.3.8 Programación Extrema

Según Kendall (2005), “la programación extrema es un enfoque para el desarrollo de software que utiliza buenas prácticas de desarrollo y las lleva a los extremos. Se basa en valores, principios y prácticas esenciales. Los cuatros valores son la comunicación, la simplicidad, la retroalimentación y la valentía. Recomendamos a los analistas de sistemas que adopten estos valores en todos los proyectos que emprendan, no sólo cuando recurran a medidas de programación extrema”.

Jeffries, Anderson y Hendrickson (2000), Es un método de desarrollo de software basado en los valores de facilidad, comunicación, retroalimentación y coraje. Funciona al involucrar a todo el equipo en un conjunto simple de prácticas y proporcionar suficiente retroalimentación para que comprendan dónde se encuentran y cómo aplicar estas prácticas a cada situación específica.

A. FASES

Según Beck, K. (1999), las fases de la Programación Extrema están altamente vinculadas y relacionadas en forma cíclica por la interacción marcada de los actores (desarrollador – cliente - usuario), estas fases son cuatro son: fase de exploración, fase de planificación, fase de iteración y la fase de producción, esta última fase puede ser incluida en la fase de iteración según la envergadura el proyecto y la decisión del equipo de desarrollo, ya que al iterar esta última fase es repetida hasta la aceptación y muerte del proyecto.

1. Exploración

Según Wesley (2003). En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de

pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

Según Joskowicz. J. (2008) Es la fase en la que se define el alcance general del proyecto. En esta fase, el cliente define lo que necesita mediante la redacción de sencillas “historias de usuarios”. Los programadores estiman los tiempos de desarrollo en base a esta información. Debe quedar claro que las estimaciones realizadas en esta fase son primarias (ya que estarán basadas en datos de muy alto nivel), y podrían variar cuando se analicen más en detalle en cada iteración.

2. Planificación

Según Joskowicz. J. (2008) La planificación es una fase corta, en la que el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario, y, asociadas a éstas, las entregas. Típicamente esta fase consiste en una o varias reuniones grupales de planificación. El resultado de esta fase es un Plan de Entregas, o “Release Plan”, como se detallará en la sección “Reglas y Practicas”.

Según Beck, K. (1999). En esta fase el cliente establece la prioridad de cada historia de usuario y la describe detalladamente correspondientemente a la regla del negocio, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días. Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la “velocidad” de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración. La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

3. Iteraciones

Según Joskowicz. J. (2008) Esta es la fase principal en el ciclo de desarrollo de XP. Las funcionalidades son desarrolladas en esta fase, generando al final de cada una un entregable funcional que implementa las historias de usuario asignadas a la iteración. Como las historias de usuario no tienen suficiente detalle como para permitir su análisis y desarrollo, al principio de cada iteración se realizan las tareas necesarias de análisis, recabando con el cliente todos los datos que sean necesarios. El cliente, por lo tanto, también debe participar activamente durante esta fase del ciclo.

Según Beck, K. (1999). Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción. Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores.

4. Puesta en producción

Según Joskowicz. J. (2008), dice que al final de cada iteración se entregan módulos funcionales y sin errores, puede ser deseable por parte del cliente no poner el sistema en producción hasta tanto no se tenga la funcionalidad completa. En esta fase no se realizan más desarrollos funcionales, pero pueden ser necesarias tareas de ajuste (“fine tuning”).

B. Diseño

La metodología XP hace especial énfasis en los diseños simples y claros, Hay que procurar hacerlo todo lo menos complicado posible para conseguir un diseño fácilmente entendible y así se podrá hacer con menos esfuerzo y poco tiempo. Los conceptos más importantes de diseño en esta metodología son los siguientes (Díaz y Collazo, 2013).

1. Simplicidad

Un diseño simple se implementa más rápidamente que uno complejo. Por ello XP propone implementar el diseño más simple posible que funcione. Se sugiere nunca adelantar la implementación de funcionalidades que no correspondan a la iteración en la que se esté trabajando. Características fundamentales del código: Testeable, legible, comprensible y explicable (Díaz y Collazo, 2013).

2. Metáforas

Una “metáfora” es algo que todos entienden, sin necesidad de mayores explicaciones. La metodología XP sugiere utilizar este concepto como una manera sencilla de explicar el propósito del proyecto, y guiar la estructura y arquitectura del mismo (Díaz y Collazo, 2013).

3. Solución “spike”

Una solución “spike”, es una solución muy simple para plantear posibles soluciones, de manera, que solamente se aborda el problema en concreto y se aísla de otro tipo de preocupaciones (Díaz y Collazo, 2013).

4. Refactorización

La recodificación consiste en escribir nuevamente parte del código de un programa, sin cambiar su funcionalidad, a los efectos de hacerlo más simple, conciso y/o entendible. Muchas veces, al terminar de escribir un código de programa, pensamos que, si lo comenzáramos de nuevo, lo hubiéramos hecho en forma diferente, más clara y eficientemente (Díaz y Collazo, 2013).

C. Roles

1. Cliente

El cliente es el responsable de conducir el proyecto. Define el proyecto y sus objetivos. Cuanto más preciso es su trabajo y cuanto mayor sea su involucración, mayores serán las oportunidades de éxito (Díaz y Collazo, 2013).

2. Programador

Una vez que se han comprendido las historias de usuario, el XP adjudica a los programadores la responsabilidad de tomar decisiones técnicas. Los desarrolladores estiman el tiempo que les va a tomar cada historia (Díaz y Collazo, 2013).

3. Encargado de Pruebas (Tester)

El encargado de pruebas ayuda al cliente a definir y escribir las pruebas de aceptación de las historias de usuario. Este rol en un equipo XP también es responsable realizar test periódicamente y informar de los resultados al equipo. A medida que el volumen de pruebas aumenta, el encargado de pruebas necesitará una herramienta para crear y mantener la batería de pruebas, ejecutarlas y obtener los resultados más rápidamente (Díaz y Collazo, 2013).

4. Encargado de Seguimiento (Tracker)

Hace el seguimiento de acuerdo a la planificación. La métrica más importante para XP es la velocidad del equipo, que se define como el tiempo ideal estimado para las tareas frente al tiempo real dedicado. Esta métrica ayuda a determinar si el proyecto está dentro del tiempo de la iteración (Díaz y Collazo, 2013).

5. Entrenador (Coach)

No es un rol cubierto en todos los equipos de XP. Su papel es guiar y orientar al equipo, especialmente cuando un equipo comienza a trabajar siguiendo la metodología XP. Esto se debe a que no es fácil aplicar XP de forma consistente. Aunque son prácticas de sentido común, se necesita un tiempo para interiorizarlas. También hay situaciones especiales en las que se requiere la sabiduría de un especialista en XP para aplicar sus normas frente a un obstáculo en el proyecto (Díaz y Collazo, 2013).

6. Gestor (Big Boss)

Es el gerente del proyecto, debe tener una idea general del proyecto y estar familiarizado con su estado. El cliente puede asumir este papel (Díaz y Collazo, 2013).

CAPÍTULO III
MATERIAL Y MÉTODOS

41.1 TIPO Y NIVEL DE INVESTIGACIÓN

41.1.1 Tipo de investigación

Según Teodoro & Nieto (2018) La investigación aplicada se centra en mejorar, perfeccionar u optimizar el funcionamiento de sistemas, procedimientos, normas y reglas tecnológicas existentes a la luz de los avances en ciencia y tecnología. Por lo tanto, este tipo de investigación no se evalúa en términos de verdadero, falso o probable, sino en términos de eficiencia, deficiencia, ineficiencia, eficacia o ineficacia.

De acuerdo a la variable identificada en el presente estudio de investigación, es de tipo **aplicada**, porque la variable solo sirvió para caracterizar una situación concreta y perfeccionar un procedimiento.

41.1.2 Nivel de investigación

Supo (2014), expone una investigación descriptiva es aquella que utiliza el método de análisis, se logra caracterizar un objeto de estudio o una situación concreta, señalando sus características y propiedades. Interpreta lo que es y describe la situación de las cosas en el presente. Combinada con ciertos criterios de clasificación, sirve para ordenar, agrupar o sistematizar los objetos involucrados en el trabajo indagatorio.

Por consiguiente esta investigacion se enmarca en un nivel descriptivo, a razón la que se busca solo caracterizar la situcion concreta, que es la implementacion de un bot en telegram, de acuerdo a los requerimientos obtenidos despues de la aplicación del instrumento de analisis documental.

41.2 DISEÑO DE LA INVESTIGACIÓN

Según Hernández, Fernández y Baptista (2010) mencionan que, las investigaciones no experimentales son aquellas que se realizan sin manipular deliberadamente las variables, es decir, no se varía intencionalmente la variable, simplemente se realiza la observación de las funciones tal y como se dan en su contexto natural para después analizarlas.

Considernado esta teoria el presente estudio tien un diseño no experimental porque no se efectua niguna manipulacion de la variable de interes para cambiar un estado situacional.

41.3 VARIABLES

41.3.1 Definición conceptual de variables

Variable de Interés

Registro de Worklogs: Es la acción sistemática mediante la cual un colaborador registra sus actividades laborales diarias dentro de una herramienta de gestión de tareas o proyectos, con el objetivo de mantener trazabilidad, control de tiempos y seguimiento de avances en el cumplimiento de objetivos.

Variables Descriptivas

Tiempo de registro: Se refiere a la duración o intervalo que toma un colaborador para registrar sus actividades laborales diarias en la plataforma correspondiente. Es un indicador de eficiencia en el proceso de documentación del trabajo.

Reporte de registro: Corresponde a la generación de informes que recopilan, organizan y presentan los datos ingresados por los colaboradores en los worklogs, permitiendo el análisis de cumplimiento, frecuencia y consistencia del registro.

41.3.2 Definición operacional de variables

Variable de Interés

Registro de Worklogs: Representa el acto de ingresar y guardar información detallada sobre las tareas realizadas por el colaborador, utilizando una interfaz digital como Jira, en un entorno controlado por la empresa.

Variables Descriptivas

Tiempo de registro: Se registra en horas y corresponde al tiempo promedio que un usuario emplea para completar el registro de sus actividades en el sistema.

Reporte de registro: Se operacionaliza a través de la visualización de informes generados automáticamente por el sistema, que consolidan los datos registrados por cada usuario en periodos determinados.

41.4 POBLACIÓN Y MUESTRA

41.4.1 Población

Según Tamayo (2012), “la población es la totalidad de un fenómeno de estudio, incluye la totalidad de unidades de análisis que integran dicho fenómeno y que debe cuantificarse para un determinado estudio integrando un conjunto N de entidades que participan de una determinada característica, y se le denomina la población por constituir la totalidad del fenómeno adscrito a una investigación” (p. 180).

En esta investigación, la población está compuesta por 49 proyectos GDN PERU NTTDATTA, 2024.

41.4.2 Muestra

En el presente estudio, se considera una muestra no probabilística por conveniencia del investigador, compuesta por 3 proyectos seleccionados de los 49 que conforman GDN NTTDATA PERÚ en el año 2024.

La elección de esta muestra se basa en criterios de disponibilidad, accesibilidad y relevancia, con el objetivo de evaluar el funcionamiento y la eficacia del bot de Telegram desarrollado para el registro de worklogs en la plataforma Jira.

41.5 TÉCNICAS E INSTRUMENTOS DE LA INVESTIGACIÓN

41.5.1 Técnicas

El análisis documental consiste en obtener un dato secundario a fin de expresar su contenido de manera correcta, es también un método para poder dar a conocer a las personas que desconocen o no les son muy entendibles la información (Chan González, 2014).

41.5.2 Instrumentos

Registro de análisis documental.

41.6 PROCEDIMIENTOS

41.6.1 Estrategia de prueba de hipótesis

En el presente estudio no se considera prueba de hipótesis por ser una investigación de nivel descriptivo que es el desarrollo de un prototipo de chat bot.

41.6.2 Técnicas de procesamiento de datos

A. Técnicas para procesamiento de datos con metodología XP

La elección de las herramientas tecnológicas que se utilizan, son seleccionados en función a las necesidades y limitaciones como: recursos económicos y recursos humanos.

Tabla 2. Herramientas para tratamiento de datos e información

NOMBRE	FABRICANTE	SERVICIO
Windows 10	Microsoft Coporation	Sistema Operativo de Microsoft línea de sistemas operativos con licencia producida por Microsoft corporación.
Visual Studio Code	Microsoft Coporation	Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft. Se destaca por su versatilidad, facilidad de uso y amplia gama de extensiones disponibles que permiten personalizar y ampliar sus funcionalidades según las necesidades del usuario.
MongoDB Compass	MongoDB, Inc	MongoDB Compass es una herramienta gráfica y visual para trabajar con bases de datos MongoDB. Proporciona una interfaz de usuario intuitiva que permite a los desarrolladores y administradores de bases de datos explorar, analizar y manipular los datos almacenados en clústeres de MongoDB de una manera más eficiente y conveniente.
MongoDB Atlas	MongoDB, Inc	MongoDB Atlas es un servicio de base de datos en la nube totalmente administrado por MongoDB, Inc. Permite a los desarrolladores implementar, administrar y escalar clústeres de bases de datos MongoDB de forma rápida y sencilla sin la necesidad de preocuparse por la configuración, mantenimiento y escalabilidad de la infraestructura subyacente.
Amazon Elastic Compute Cloud (EC2)	AWS	Es un servicio de cómputo en la nube que forma parte de Amazon Web Services (AWS). Permite a los usuarios alquilar máquinas virtuales (instancias) en la nube bajo demanda, lo que les proporciona una capacidad informática escalable y flexible.
JavaScript	JavaScript	JavaScript es un lenguaje de programación ampliamente utilizado, especialmente en el desarrollo web. A menudo se utiliza para agregar interactividad y funcionalidad dinámica a las páginas web.
NODEJS	NodeJs	Node.js es un entorno de ejecución de código abierto basado en el motor de JavaScript V8 de Google Chrome. Permite a los desarrolladores ejecutar código JavaScript del lado del servidor, lo que les permite construir aplicaciones web y servicios de red altamente escalables y eficientes.
API de Telegram	Telegram	Es una interfaz de programación de aplicaciones (API) que permite a los desarrolladores interactuar con la plataforma de mensajería Telegram para crear bots, aplicaciones y servicios que utilicen su infraestructura y funcionalidades.

Fuente: Elaboración propia.

Tabla 3. Técnicas para aplicar la programación extrema, fase de exploración

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLES
Escribir historias de usuario	Historia de usuario	Describir brevemente la historia de usuario con la regla del negocio (lo que el sistema debe hacer) Dividir historias de usuario grandes	Cliente
Probar las tecnologías a utilizar	Arquitectura técnica inicial	Explorar posibilidades de uso de tecnologías Probar el rendimiento de las tecnologías Definir las tecnologías a usar	Cliente Programador Entrenador
Estimar esfuerzo para historia de usuario	**Plan de alto nivel**	**Conocer previamente la historia de usuario Hacer una implementación rápida de historia de usuario Estimar esfuerzo (semana) para desarrollar la historia de usuario**	**Programador**

Fuente: Porras (2010).

Tabla 4. Técnicas para aplicar la programación extrema, fase de planificación

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLES
Rescribir las historias de usuario	Historia de usuario	Describir detalladamente la historia de usuario con la regla de negocio	Cliente
Formular el plan de versiones	Plan de versión (una iteración)	Introducir nuevos requisitos del software Definir prioridad para cada historia de usuario por necesidad del negocio	Cliente
		Utilizar técnicas de elaboración del plan de alto nivel Estimar y asignar esfuerzo (semana) para cada historia de usuario en función a tiempo para planear, diseñar, implementar y probar Estimar y asignar riesgo a cada historia de usuario en función a situación que afecta la estimación del esfuerzo Actualizar tarjeta de historia de usuario	**Programador**

Fuente: Porras (2010).

Tabla 5. Técnicas para aplicar la programación extrema, fase de iteración

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLES
Definir la	Arquitectura	Actualizar la arquitectura técnica	Cliente

arquitectura técnica	técnica	inicial Usar características del negocio Utilizar arquitectura por capas Integrar frameworks	Programador Entrenador
Escribir tareas de ingeniería	Tarea de ingeniería	Dividir cada historia de usuario en tareas, describir usando reglas del negocio cada tarea de ingeniería	Cliente programador
Formular el plan de iteraciones	Plan de iteración	Estimar y asignar esfuerzo para desarrollar una tarea de ingeniería	Programador
		Asignar una tarea de ingeniería al programador	Entrenador Programador
		Utilizar el plan de versión Actualizar el plan con tareas de ingeniería de la siguiente iteración Actualizar las historias de usuario Actualizar el plan con tareas no concluidas Actualizar las tarjetas de tarea de ingeniería	
Implementar las interfaces	GUI	Diseñar con precisión la GUI relacionada a cada historia de usuario Generar código para la interface usando herramienta	Cliente Programador
Escribir tarjetas CRC para cada tarea de ingeniería	Tarjeta CRC	Diseñar para una tarea de ingeniería de forma simple Rediseñar por falla de prueba de aceptación una tarea Identificar responsabilidades Identificar colaboración Identificar Atributos	Cliente Programador
Implementar la base de datos física	Base de datos física	Escribir script usando tarjeta CRC Ejecutar script usando DBMS	Programador
Implementar código para clases entidad	Código fuente	Escribir código fuente o generar con una herramienta usando tarjetas CRC	Programador
Crear pruebas unitarias para las clases control	Prueba unitaria	Escribir código fuente para una prueba unitaria, usando una herramienta	Programador
Implementar código fuente	Código fuente	Codificar una tarea de ingeniería Hacer refactoring	Programador Supervisor

		Mover programadores	
Ejecutar pruebas unitarias	Reporte de prueba unitaria	Ejecutar el módulo de cada prueba unitaria Modificar código fuente si la prueba unitaria muestra resultado incorrecto	Programador
Realizar integración continua	Código fuente	Integrar las tareas para una historia de usuario Mantener sistema integrado todo el tiempo	Programador
Ejecutar pruebas de integración para una historia de usuario	Reporte pruebas de integración	Integrar continuamente al concluir las tareas de una historia de usuario Verificar que las pruebas de integración pasan al 100%	Programador
Ejecutar pruebas de aceptación	Reporte de pruebas de aceptación	Correr la última versión de una iteración Utilizar los casos de prueba de aceptación	Cliente Encargado de pruebas

Fuente: Porras (2010).

B. Técnicas de análisis de datos

Los datos que se obtienen mediante los instrumentos; Análisis documental y Registro de análisis documental, se analizan para procesarlo usando el marco de trabajo xp, a fin de desarrollar los procesos en estudio que nos presentaran información sobre; datos de tiempo de registro y reporte de registro.

41.6.3 Diseño estadístico

De acuerdo al nivel de investigación que es descriptiva no se presenta el diseño para el contraste de hipótesis. Los resultados del prototipo de software se presentarán mediante tablas y figuras.

CAPÍTULO IV
RESULTADOS Y DISCUSIÓN

42.1 RESULTADOS

Aplicando la metodología de programación extrema y según sus fases, se muestra a continuación los resultados de cada fase.

FASE DE EXPLORACIÓN

De acuerdo a los procedimientos en la metodología de programación extrema la fase uno: Exploración, se procedió a recopilar las necesidades iniciales mediante reuniones y entrevistas con los colaboradores, durante esta esta etapa, se identificó la problemática principal relacionada con el registro poco eficiente de worklogs en jira, lo cual generaba retrasos, falta de control y baja visibilidad sobre las tareas diarias. A partir de ello, se formularon las historias de usuario y la arquitectura técnica inicial, como se muestra a continuación.

Historias de Usuario

Tabla 6. Historia de Usuario

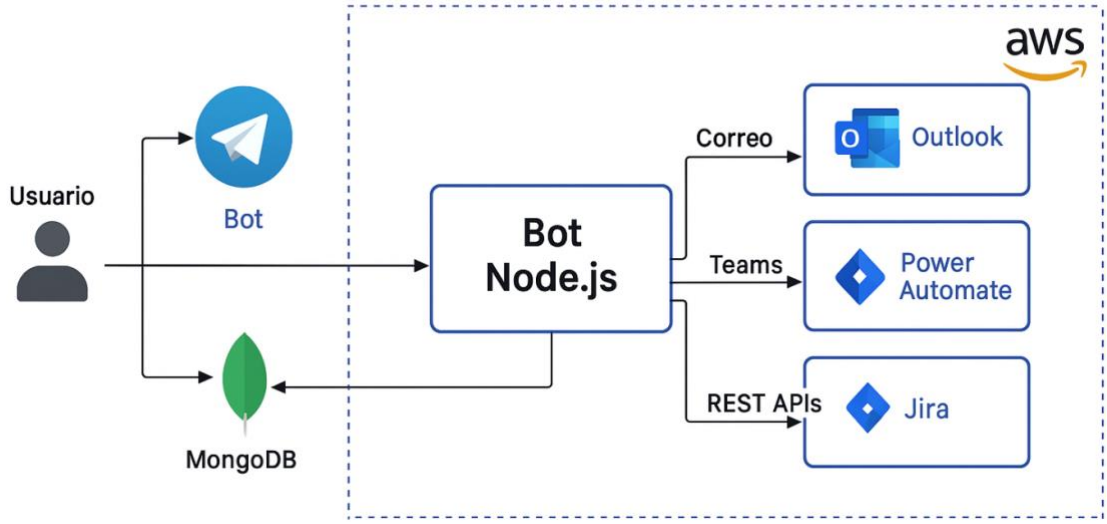
N°	HISTORIA DE USUARIO	DESCRIPCIÓN
1	Autenticación en el bot	Como colaborador de NTT DATA, quiero autenticarme en el bot de Telegram mediante un código enviado a mi correo corporativo, para acceder a funcionalidades internas de la empresa.
2	Consultar mi información	Como colaborador, quiero consultar mi información personal desde el bot, para ver detalles de mi perfil, estado y asignaciones activas.
3	Actualizar información masiva	Como administrador del bot, quiero actualizar la información de usuarios a través del envío de un archivo Excel, para mantener actualizados los datos personales y asignaciones.

4	Consultar información de otros usuarios	Como administrador del bot, quiero consultar la información personal de cualquier colaborador mediante el comando /persona, para ver su estado, rol, y asignaciones actuales.
5	Visualización de tareas	Como usuario autenticado, quiero ver la lista de tareas asignadas desde el bot, para dar seguimiento al estado y progreso de cada una.
6	Registrar tiempo en tarea	Como usuario autenticado, quiero registrar tiempo trabajado en una tarea, para dejar constancia diaria de mi avance.
7	Cambiar estado de tarea	Como usuario autenticado, quiero cambiar el estado de una tarea seleccionada, para reflejar su avance en base a los estados válidos disponibles para esa tarea.
8	Ver historial de incurridos	Como usuario autenticado, quiero consultar el historial de incurridos de una tarea, para ver qué usuarios han registrado horas y en qué fechas.
9	Reporte mensual (SMS)	Como usuario autenticado, quiero visualizar un resumen de mis incurridos por día del mes en formato mensaje, para consultar rápidamente desde el bot si tengo horas pendientes.
10	Reporte mensual (Excel)	Como usuario, quiero descargar un archivo Excel con el detalle de mis incurridos del mes, para revisar mi carga horaria con mayor precisión y respaldarla si es necesario.
11	Reporte por comando /worklog	Como usuario autenticado, quiero generar un reporte de incurridos por rango de fecha usando el comando /worklog, para obtener un archivo Excel con mis registros detallados.
12	Reporte por usuarios múltiples	Como jefe de equipo, quiero generar reportes de incurridos para varios usuarios especificando sus nombres en el comando /worklog, para tener control sobre los registros de mi equipo.
13	Registrar incurrido masivo	Como usuario autenticado, quiero registrar múltiples incurridos de manera masiva usando un archivo CSV, para agilizar la carga de tareas realizadas en distintos días.
14	Incurrir días festivos	Como usuario autenticado, quiero registrar automáticamente mis horas en todos los días festivos del año para un issueKey específico, para ahorrar tiempo y mantener un registro uniforme para mi equipo.
15	Reportes del equipo (líder)	Como jefe de equipo, quiero generar un reporte de incurridos de todos los miembros de mi equipo, para supervisar su avance diario o por rango de fechas.

16	Alerta de no incurridos	Como jefe de equipo, quiero enviar alertas a los miembros de mi equipo que no han registrado incurrido en el día, para recordarles y evitar omisiones.
17	Notificación automática diaria	Como jefe de equipo, quiero que el bot ejecute automáticamente una revisión diaria de incurridos y notifique por Teams a los miembros que no han registrado horas, para garantizar cumplimiento sin intervención manual.

Figura 2

Arquitectura Técnica Inicial



PLAN DE ALTO NIVEL

Tabla 7. Lista de historia de usuario

N°	HISTORIA DE USUARIO	ESFUERZO ESTIMADO
1	Autenticación en el bot	2 días
2	Consultar mi información	1 día
3	Actualizar información masiva	2 días
4	Consultar información de otros usuarios	1 día
5	Visualización de tareas	2 días
6	Registrar tiempo en tarea	2 días
7	Cambiar estado de tarea	1 día
8	Ver historial de incurridos	1 día
9	Reporte mensual (SMS)	1 día
10	Reporte mensual (Excel)	2 días
11	Reporte por comando /worklog	2 días
12	Reporte por usuarios múltiples	2 días
13	Registrar incurrido masivo	2 días

14	Incurrir días festivos	2 días
15	Reportes del equipo (líder)	2 días
16	Alerta de no incurridos	2 días
17	Notificación automática diaria	2 días

FASE DE PLANIFICACIÓN

En esta etapa, se procedió a definir los artefactos fundamentales del proyecto mediante el uso de historias de usuario previamente levantadas durante la fase exploratoria. En esta etapa se estimó el esfuerzo requerido para cada historia utilizando criterios de complejidad técnica, integración con sistemas externos y validaciones internas. Se asigno un esfuerzo en días a cada funcionalidad para permitir su agrupación en iteraciones de desarrollo.

Se construyó un plan de versión inicial organizando las historias de usuario en bloques funcionales distribuidos en cuatro iteraciones. Esta planificación permitió una gestión eficiente del tiempo, para así asegurar los entregables de manera progresiva y funcionales a lo largo del desarrollo del sistema.

Tabla 8. Autenticación en el bot

Historia de Usuario	
Número: 1	**Usuario:** Colaborador
Nombre historia: Autenticación en el bot	
Prioridad en negocio: Alto	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Programador principal	
Descripción: El colaborador accede al bot mediante código enviado por correo, asegurando su autenticidad para habilitar las funciones internas.	
Observaciones: Ninguno	

Tabla 9. Consultar mi información

Historia de Usuario	
Número: 2	**Usuario:** Colaborador
Nombre historia: Consultar mi información	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Programador principal	
Descripción: Permite al colaborador consultar su información personal, estado y asignaciones actuales.	
Observaciones: Ninguno	

Tabla 10. Actualizar información masiva

Historia de Usuario	
Número: 3	**Usuario:** Administrador
Nombre historia: Actualizar información masiva	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Programador principal	
Descripción: Carga y actualización de datos de múltiples usuarios mediante archivo Excel.	
Observaciones: Formato debe de validarse antes de guardar	

Tabla 11. Consultar información de otros usuarios

Historia de Usuario	
Número: 4	**Usuario:** Administrador
Nombre historia: Consultar información de otros usuarios	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Programador principal	
Descripción: Permite consultar el perfil y asignaciones de otro usuario a través del comando /persona.	
Observaciones: Acceso restringido solo administradores.	

Tabla 12. Visualización de tareas

Historia de Usuario	
Número: 5	**Usuario:** Colaborador
Nombre historia: Visualización de tareas	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Programador principal	
Descripción: El usuario puede visualizar sus tareas activas en Jira con estado, progreso y tiempo estimado.	
Observaciones: Ninguno.	

Tabla 13. Registrar tiempo tarea

Historia de Usuario	
Número: 6	**Usuario:** Colaborador
Nombre historia: Registrar tiempo tarea	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: Programador principal	

Descripción: El colaborador registra el tiempo trabajado por día en una tarea específica.
Observaciones: Registrar en el formato h, d, m.

Tabla 14. Cambiar estado de tarea

Historia de Usuario	
Número: 7	**Usuario:** Colaborador
Nombre historia: Cambiar estado de tarea	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Programador principal	
Descripción: Permite cambiar el estado de una tarea según los valores válidos disponibles.	
Observaciones: Validar transición permitida.	

Tabla 15. Ver historial de incurridos

Historia de Usuario	
Número: 8	**Usuario:** Colaborador
Nombre historia: Ver historial de incurridos	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Programador principal	
Descripción: Visualiza las horas registradas en una tarea por día y por usuario.	
Observaciones: Ninguno.	

Tabla 16. Reporte mensual (SMS)

Historia de Usuario	
Número: 9	**Usuario:** Colaborador
Nombre historia: Reporte mensual (SMS)	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Programador principal	
Descripción: Muestra al usuario un resumen diario del mes con sus incurridos directamente en el bot.	
Observaciones: El resultado debe mostrarse de forma ordenada	

Tabla 17. Reporte mensual (Excel)

Historia de Usuario	
Número: 10	**Usuario:** Colaborador

Nombre historia: Reporte mensual (Excel)	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: Programador principal	
Descripción: Genera un archivo Excel con los días, horas registradas, esperadas y faltantes.	
Observaciones: Ninguno	

Tabla 18. Reporte por comando "/worklog"

Historia de Usuario	
Número: 11	**Usuario:** Colaborador
Nombre historia: Reporte por comando "/worklog"	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 3
Programador responsable: Programador principal	
Descripción: Permite generar reportes por fecha usando el comando "/worklog" con exportación a Excel.	
Observaciones: Validar parámetros de entrada.	

Tabla 19. Reporte por usuarios múltiples

Historia de Usuario	
Número: 12	**Usuario:** Lider de equipo
Nombre historia: Reporte por usuarios múltiples	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 3
Programador responsable: Programador principal	
Descripción: Permite al líder generar un Excel de varios usuarios ingresados separados por coma.	
Observaciones: Ninguno.	

Tabla 20. Registrar incurrido masivo (CSV)

Historia de Usuario	
Número: 13	**Usuario:** Colaborador
Nombre historia: Registrar incurrido masivo (CSV)	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 2	**Iteración asignada:** 3
Programador responsable: Programador principal	
Descripción: Carga de múltiples registros de tareas por fechas mediante CSV.	
Observaciones: El formato debe ser validado.	

Tabla 21. Incurrir días festivos

Historia de Usuario	
Número: 14	**Usuario:** Colaborador
Nombre historia: Incurrir días festivos	
Prioridad en negocio: Medio	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 3
Programador responsable: Programador principal	
Descripción: Permite registrar automáticamente los días festivos del año usando un issueKey.	
Observaciones: Festivos provienen de base de datos.	

Tabla 22. Reportes del equipo (líder)

Historia de Usuario	
Número: 15	**Usuario:** Lider de equipo
Nombre historia: Reportes del equipo (líder)	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 4
Programador responsable: Programador principal	
Descripción: Muestra los incurridos por todo el equipo de trabajo agrupado por mes o por rango.	
Observaciones: Requiere validación de rol.	

Tabla 23. Alerta de no incurridos

Historia de Usuario	
Número: 16	**Usuario:** Lider de equipo
Nombre historia: Alerta de no incurridos	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 4
Programador responsable: Programador principal	
Descripción: Envía recordatorios a los usuarios del equipo que no registraron horas en el día.	
Observaciones: Se ejecuta bajo demanda.	

Tabla 24. Notificación **automática** diaria

Historia de Usuario	
Número: 17	**Usuario:** Sistema
Nombre historia: Notificación automática diaria	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 4
Programador responsable: Programador principal	

Descripción: Scheduler automatizado ejecutado diariamente a las 17:30 para verificar worklogs y enviar alertas por Teams.
Observaciones: Depende del cron del servidor.

Plan de Versión inicial

Tabla 25. Lista de Plan de versión inicial

Versión	Duración estimada	Historias de Usuario (HU)	Objetivo / Entregables
Versión 1.0	8 días	HU01, HU02, HU03, HU04, HU05	Módulo de autenticación, consulta de perfil, carga masiva y visualización de tareas.
Versión 1.1	8 días	HU06, HU07, HU08, HU09, HU10	Funciones operativas: registrar tiempo, cambiar estado, worklog y reportes individuales.
Versión 1.2	8 días	HU11, HU12, HU13, HU14	Automatización y eficiencia: comando /worklog, múltiples usuarios, festivos y CSV masivo.
Versión 1.3	7 días	HU15, HU16, HU17	Funciones para líderes: reportes de equipo, alertas y scheduler de Teams.

Planificación por iteraciones.

Tabla 26. Lista de Planificación por iteraciones

Iteración	Historias Incluidas	Duración Estimada (días)
Iteración 1	HU01, HU02, HU03, HU04, HU05	8
Iteración 2	HU06, HU07, HU08, HU09, HU10	8
Iteración 3	HU11, HU12, HU13, HU14	8
Iteración 4	HU15, HU16, HU17	7

FASE DE ITERACIÓN

La etapa iterativa permite obtener entregables: arquitectura técnica, tareas de ingeniería, plan de iteración, casos de prueba de aceptación, GUI, tarjeta CRC, base de datos física, el código fuente de la clase de la entidad, el código fuente de la tarea de ingeniería, el informe de prueba de la unidad, el informe de prueba de integración y la aceptación son los siguientes.

Figura 3

Arquitectura tecnica final – Diagrama de despliegue

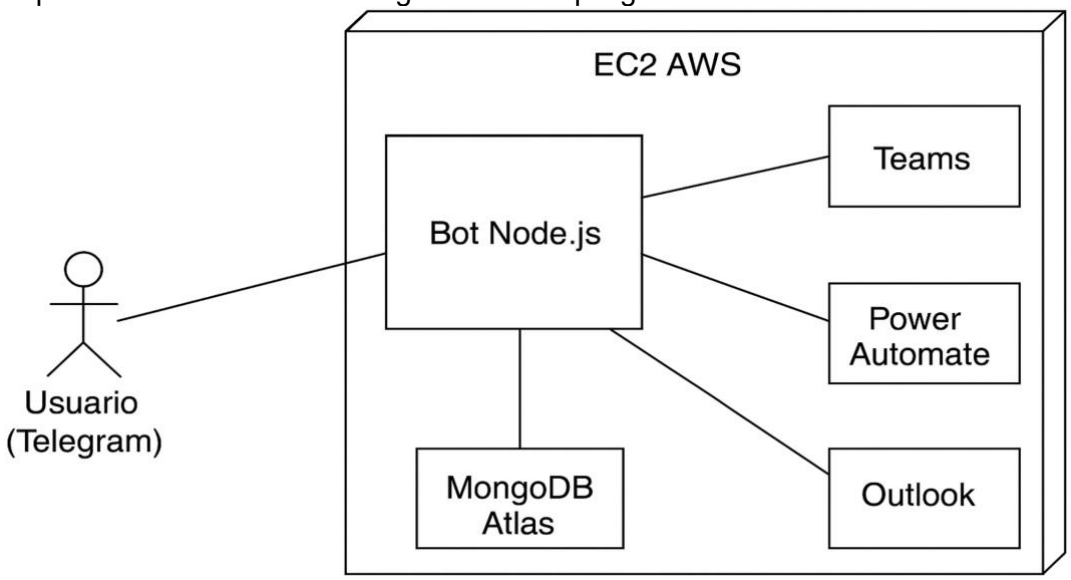


Figura 4

Arquitectura tecnica dinal - Diagrama de componentes

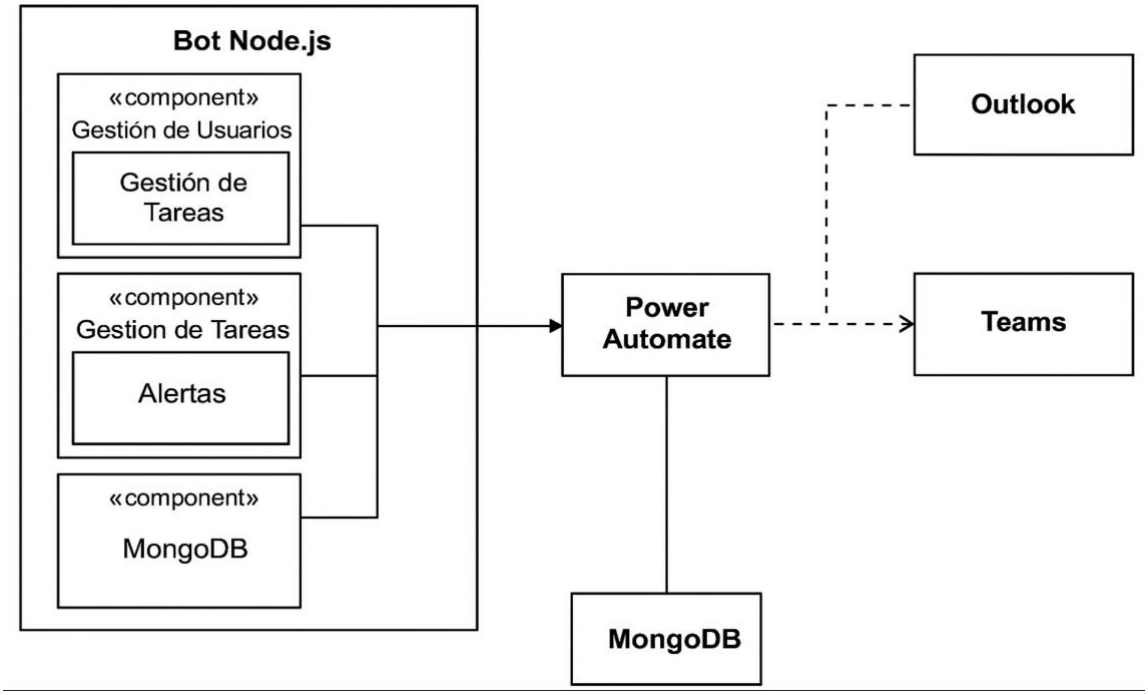


Tabla 27. Autenticación de usuario vía código único

TAREA DE INGENIERÍA	
Número tarea de ingeniería: 1	**Número historia de usuario:** 1
Nombre tarea: Autenticación de usuario vía código único	
Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 01/04/2025	Fecha fin: 03/04/2025

Programador responsable: Desarrollador Principal
Descripción: El bot solicita el ingreso del usuario corto de NTT DATA. Posteriormente, envía automáticamente un código de verificación al correo asociado. El usuario debe ingresar ese código en Telegram para completar su autenticación. Al validarse correctamente, el bot habilita el menú principal de opciones disponibles.
Observaciones: Solo usuarios pertenecientes a la empresa se podrán registrar.

Tabla 28. Consultar mi información

TAREA DE INGENIERÍA	
Número tarea de ingeniería: 2	**Número historia de usuario:** 2
Nombre tarea: Consultar mi información	
Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 04/04/2025	Fecha fin: 05/04/2025
Programador responsable: Desarrollador Principal	
Descripción: El usuario autenticado puede consultar su información personal almacenada en MongoDB. Los datos incluyen: código, nombre completo, rol, fecha de ingreso, estado (activo/inactivo) y asignaciones actuales (cliente, equipo, fechas de inicio y fin, tecnología).	
Observaciones: Validar que solo usuarios autenticados puedan acceder a su información. Mostrar la información de forma ordenada y clara.	

Tabla 29. Carga y actualización masiva de información de usuarios

TAREA DE INGENIERÍA	
Número tarea de ingeniería: 3	**Número historia de usuario:** 3
Nombre tarea: Carga y actualización masiva de información de usuarios	
Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 06/04/2025	Fecha fin: 08/04/2025
Programador responsable: Desarrollador Principal	
Descripción: El administrador puede actualizar la información de usuarios mediante el comando /people que permite subir un archivo Excel. Este archivo contiene la información personal y asignaciones actualizadas de los usuarios. El bot procesa el archivo y actualiza los datos en MongoDB.	
Observaciones: Validar que el archivo cumpla con la estructura y columnas requeridas. Permitir solo a administradores autorizados realizar esta acción.	

Tabla 30. Consulta de información de otros usuarios mediante comando

TAREA DE INGENIERÍA	
Número tarea de ingeniería: 4	**Número historia de usuario:** 4
Nombre tarea: Consulta de información de otros usuarios mediante comando	
Tipo de tarea: Desarrollo	**Puntos estimados**: 1
Fecha inicio: 09/04/2025	Fecha fin: 10/04/2025
Programador responsable: Desarrollador Principal	
Descripción: El administrador puede consultar la información personal y asignaciones activas de cualquier usuario mediante el comando /persona username. El bot responde con el nombre, código, rol, cliente asignado, equipo y estado actual.	
Observaciones: Validar que solo los perfiles con rol "ADMIN" puedan ejecutar esta función. Manejar errores si el usuario no existe o está inactivo.	

Tabla 31. **Visualización** de tareas asignadas al usuario

TAREA DE INGENIERÍA	
Número tarea de ingeniería: 5	**Número historia de usuario:** 5
Nombre tarea: Visualización de tareas asignadas al usuario	
Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 11/04/2025	Fecha fin: 12/04/2025
Programador responsable: Desarrollador Principal	
Descripción: El bot consulta directamente a Jira las tareas asignadas al usuario autenticado. Lista las tareas activas mostrando el código (issueKey), título, estado actual y posibles acciones disponibles (Incurrir, Cambiar estado, Ver Worklog).	
Observaciones: Optimizar el listado para que sea claro y navegable desde Telegram. Manejar casos donde no haya tareas asignadas.	

Tabla 32. Registrar tiempo trabajado en tarea (incurrido)

TAREA DE INGENIERÍA	
Número tarea de ingeniería: 6	**Número historia de usuario:** 6
Nombre tarea: Registrar tiempo trabajado en tarea (incurrido)	
Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 13/04/2025	Fecha fin: 15/04/2025
Programador responsable: Desarrollador Principal	
Descripción: El usuario autenticado puede registrar el tiempo trabajado en una tarea seleccionada de Jira. El bot solicita el tiempo trabajado (en formato h, m o d) y un comentario opcional. El registro se envía mediante la API de Jira y se almacena directamente en la tarea seleccionada.	

Observaciones: Validar que el tiempo ingresado sea positivo y que el registro se asocie a la fecha actual. Manejar errores si el issueKey no acepta worklogs.

Tabla 33. Cambiar estado de tarea

TAREA DE INGENIERÍA	
Número tarea de ingeniería: 7	Número historia de usuario: 7
Nombre tarea: Cambiar estado de tarea	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha inicio: 16/04/2025	Fecha fin: 17/04/2025
Programador responsable: Desarrollador Principal	
Descripción: El usuario autenticado puede cambiar el estado de una tarea seleccionada. El bot consulta los estados permitidos para esa tarea a través de la API de Jira y muestra las opciones disponibles. El cambio de estado se aplica directamente en Jira tras la selección del usuario.	
Observaciones: Validar que solo se muestren estados válidos y permitidos por el flujo de trabajo (workflow) configurado en Jira. Registrar el cambio en los logs del bot.	

Tabla 34. Consulta de historial de incurridos (worklogs) de una tarea

TAREA DE INGENIERÍA	
Número tarea de ingeniería: 8	**Número historia de usuario:** 8
Nombre tarea: Consulta de historial de incurridos (worklogs) de una tarea	
Tipo de tarea: Desarrollo	**Puntos estimados**: 1
Fecha inicio: 18/04/2025	Fecha fin: 19/04/2025
Programador responsable: Desarrollador Principal	
Descripción: El usuario puede consultar el historial de incurridos registrados en una tarea seleccionada. El bot consulta a Jira y muestra una lista de registros con usuario, fecha y tiempo trabajado.	
Observaciones: Asegurar que la información se muestre de manera clara y ordenada. Incluir control de acceso: solo el usuario que registra o administradores pueden ver el historial completo.	

Tabla 35. Generación de reporte mensual en formato mensaje (SMS)

TAREA DE INGENIERÍA	
Número tarea de ingeniería: 9	**Número historia de usuario:** 9
Nombre tarea: Generación de reporte mensual en formato mensaje (SMS)	
Tipo de tarea: Desarrollo	**Puntos estimados**: 1
Fecha inicio: 20/04/2025	Fecha fin: 21/04/2025
Programador responsable: Desarrollador Principal	

Descripción: El bot genera un resumen diario de los incurridos del mes actual y lo presenta al usuario en formato de mensaje dentro de Telegram. Muestra días con worklogs registrados y días faltantes.
Observaciones: Asegurar formato claro y fácil de leer desde dispositivos móviles. Validar que la información se extraiga directamente desde Jira sin usar la base de datos local.

Tabla 36. Generación de reporte mensual en formato Excel

TAREA DE INGENIERÍA	
Número tarea de ingeniería: 10	**Número historia de usuario:** 10
Nombre tarea: Generación de reporte mensual en formato Excel	
Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 22/04/2025	Fecha fin: 24/04/2025
Programador responsable: Desarrollador Principal	
Descripción: El bot permite al usuario descargar un archivo Excel con el detalle de sus incurridos del mes. El archivo incluye fechas, tareas (issueKey), horas registradas, horas esperadas y comentarios opcionales.	
Observaciones: Verificar que el archivo tenga el formato estándar utilizado por el equipo y que la información provenga directamente de Jira. Incluir validaciones para evitar exportaciones vacías.	

Tabla 37. Generación de reporte de incurridos mediante comando /worklog

TAREA DE INGENIERÍA	
Número tarea de ingeniería: 11	**Número historia de usuario:** 11
Nombre tarea: Generación de reporte de incurridos mediante comando /worklog	
Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 25/04/2025	Fecha fin: 26/04/2025
Programador responsable: Desarrollador Principal	
Descripción: El usuario autenticado puede generar un reporte detallado de sus incurridos dentro de un rango de fechas especificado usando el comando /worklog fechaInicio,fechaFin. El resultado se entrega como archivo Excel.	
Observaciones: Validar el formato correcto de fechas ingresadas (yyyy-mm-dd). Controlar que las fechas no excedan periodos mayores a un mes sin confirmación adicional. Extraer los datos directamente desde Jira.	

Tabla 38. Generación de reporte de incurridos para múltiples usuarios.

TAREA DE INGENIERÍA	
Número tarea de ingeniería: 12	**Número historia de usuario:** 12
Nombre tarea: Generación de reporte de incurridos para múltiples usuarios	
Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 27/04/2025	Fecha fin: 29/04/2025
Programador responsable: Desarrollador Principal	
Descripción: El líder de equipo puede generar un reporte consolidado de incurridos para varios usuarios mediante el comando /worklog fechaInicio,fechaFin usuario1,usuario2, … El bot devuelve un archivo Excel con el detalle de incurridos de los usuarios especificados.	
Observaciones: Validar que los usuarios indicados pertenezcan al equipo del líder que ejecuta el comando. Controlar que el número de usuarios no exceda un límite razonable (ej. máximo 10 usuarios por reporte para evitar sobrecarga).	

Tabla 39. Registro masivo de incurridos mediante archivo CSV.

TAREA DE INGENIERÍA	
Número tarea de ingeniería: 13	**Número historia de usuario:** 13
Nombre tarea: Registro masivo de incurridos mediante archivo CSV	
Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 30/04/2025	Fecha fin: 01/05/2025
Programador responsable: Desarrollador Principal	
Descripción: El líder de equipo puede generar un reporte consolidado de incurridos para varios usuarios mediante el comando /worklog fechaInicio,fechaFin usuario1,usuario2, … El bot devuelve un archivo Excel con el detalle de incurridos de los usuarios especificados.	
Observaciones: Validar que los usuarios indicados pertenezcan al equipo del líder que ejecuta el comando. Controlar que el número de usuarios no exceda un límite razonable (ej. máximo 10 usuarios por reporte para evitar sobrecarga).	

Tabla 40. Registro automático de incurridos en días festivos

TAREA DE INGENIERÍA	
Número tarea de ingeniería: 14	**Número historia de usuario:** 14
Nombre tarea: Registro automático de incurridos en días festivos	
Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 02/05/2025	Fecha fin: 03/05/2025
Programador responsable: Desarrollador Principal	

Descripción: El usuario puede registrar automáticamente incurridos en todos los días festivos del año para un issueKey específico. Los días festivos se obtienen de la base de datos MongoDB. Los registros se envían directamente a Jira.
Observaciones: Validar que los días festivos estén actualizados en la base de datos antes de proceder. Notificar al usuario si alguna fecha ya cuenta con incurridos para evitar duplicados.

Tabla 41. Generación de reportes de incurridos para el equipo (líder)

TAREA DE INGENIERÍA	
Número tarea de ingeniería: 15	**Número historia de usuario:** 15
Nombre tarea: Generación de reportes de incurridos para el equipo (líder)	
Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 04/05/2025	Fecha fin: 05/05/2025
Programador responsable: Desarrollador Principal	
Descripción: El líder de equipo puede generar reportes de incurridos de todos los miembros de su equipo por mes o por rango de fechas. El reporte puede ser recibido en formato Excel o mensaje (SMS) dentro de Telegram.	
Observaciones: Controlar que el líder solo pueda acceder a los datos de los usuarios bajo su responsabilidad. Validar la integridad de los datos exportados y que se obtengan directamente desde Jira.	

Tabla 42. Envío de alertas de no incurridos a miembros de equipo

TAREA DE INGENIERÍA	
Número tarea de ingeniería: 16	**Número historia de usuario:** 16
Nombre tarea: Envío de alertas de no incurridos a miembros de equipo	
Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 06/05/2025	Fecha fin: 07/05/2025
Programador responsable: Desarrollador Principal	
Descripción: El líder de equipo puede enviar manualmente alertas a los miembros de su equipo que no hayan registrado incurrido en el día. Las alertas se envían a través de Power Automate, que notifica por Teams o correo Outlook.	
Observaciones: Validar que solo los usuarios activos sean notificados. Excluir automáticamente los días no laborables y festivos. Controlar que la función solo esté disponible para usuarios con perfil de líder de equipo.	

Tabla 43. Scheduler de notificaciones automáticas diarias

TAREA DE INGENIERÍA	
Número tarea de ingeniería: 17	**Número historia de usuario:** 17
Nombre tarea: Scheduler de notificaciones automáticas diarias	
Tipo de tarea: Desarrollo	**Puntos estimados**: 2
Fecha inicio: 08/05/2025	Fecha fin: 09/05/2025
Programador responsable: Desarrollador Principal	
Descripción: El sistema ejecuta automáticamente todos los días a las 17:30 un proceso que identifica los usuarios que no han registrado incurridos en la fecha actual. Envía una notificación grupal al canal de Teams de cada equipo y un mensaje individual a cada usuario, utilizando Power Automate.	
Observaciones: Validar que el scheduler excluya sábados, domingos y festivos. Registrar logs diarios de ejecución y notificaciones enviadas.	

PLAN DE ITERACIÓN (Primera)

Tabla 44. Lista de Plan de Iteración (Primera)

Nº	HISTORIA DE USUARIO	TAREA DE INGENIERÍA
1	HU1 - Autenticación en el bot	Configuración de autenticación en el bot
2	HU2 - Consultar mi información	Implementación de consulta de información personal
3	HU3 - Actualizar información masiva	Desarrollo de carga masiva de información (Excel)
4	HU4 - Consultar información de otros usuarios	Implementar consulta de información por administrador
5	HU5 - Visualización de tareas	Visualización de tareas asignadas

PLAN DE ITERACIÓN (Segunda)

Tabla 45. Lista de Plan de Iteración (Segunda)

Nº	HISTORIA DE USUARIO	TAREA DE INGENIERÍA
6	HU6 - Registrar tiempo en tarea	Registro de tiempo (incurrido) en tareas
7	HU7 - Cambiar estado de tarea	Cambiar estado de tareas Jira
8	HU8 - Ver historial de incurridos	Consulta de historial de worklogs
9	HU9 - Reporte mensual (SMS)	Generar reporte mensual en formato SMS
10	HU10 - Reporte mensual (Excel)	Generar reporte mensual en formato Excel

PLAN DE ITERACIÓN (Tercera)

Tabla 46. Lista de Plan de Iteración (Tercera)

Nº	HISTORIA DE USUARIO	TAREA DE INGENIERÍA
11	HU11 - Reporte por comando /worklog	Generar reporte por rango de fechas

12	HU12 - Reporte por usuarios múltiples	Generar reporte para múltiples usuarios
13	HU13 - Registrar incurrido masivo	Registro masivo de incurridos (CSV)
14	HU14 - Incurrir días festivos	Registro automático en días festivos

PLAN DE ITERACIÓN (Cuarta)

Tabla 47. Lista de Plan de Iteración (Cuarta)

Nº	HISTORIA DE USUARIO	TAREA DE INGENIERÍA
15	HU15 - Reportes del equipo (líder)	Generación de reportes de equipo
16	HU16 - Alerta de no incurridos	Envío de alertas a miembros sin incurridos
17	HU17 - Notificación automática diaria	Desarrollo del scheduler para alertas automáticas

Plan de iteración clasificados en fechas de desarrollo

Tabla 48. Lista de iteración clasificados en fechas de desarrollo

Nº	TAREA DE INGENIERÍA	FECHA INICIO	FECHA FIN	RESPONSABLE
1	1	01/04/2025	02/04/2025	Desarrollador Principal
2	2	03/04/2025	04/04/2025	Desarrollador Principal
3	3	05/04/2025	06/04/2025	Desarrollador Principal
4	4	07/04/2025	08/04/2025	Desarrollador Principal
5	5	09/04/2025	10/04/2025	Desarrollador Principal
6	6	11/04/2025	12/04/2025	Desarrollador Principal
7	7	13/04/2025	14/04/2025	Desarrollador Principal
8	8	15/04/2025	16/04/2025	Desarrollador Principal
9	9	17/04/2025	18/04/2025	Desarrollador Principal
10	10	22/04/2025	24/04/2025	Desarrollador Principal
11	11	25/04/2025	26/04/2025	Desarrollador Principal
12	12	27/04/2025	29/04/2025	Desarrollador Principal
13	13	30/04/2025	01/06/2025	Desarrollador Principal
14	14	02/05/2025	03/05/2025	Desarrollador Principal
15	15	04/05/2025	05/05/2025	Desarrollador Principal
16	16	06/05/2025	07/05/2025	Desarrollador Principal
17	17	08/05/2025	09/05/2025	Desarrollador Principal

Interfaces

Figura 5

Inicio del bot, login en el bot

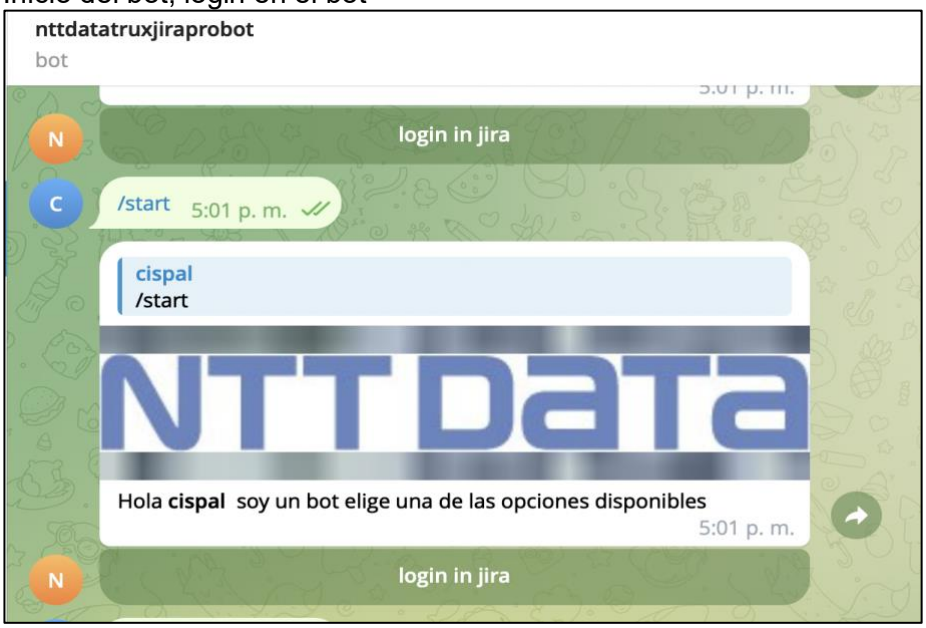


Figura 6

Envío código de verificación

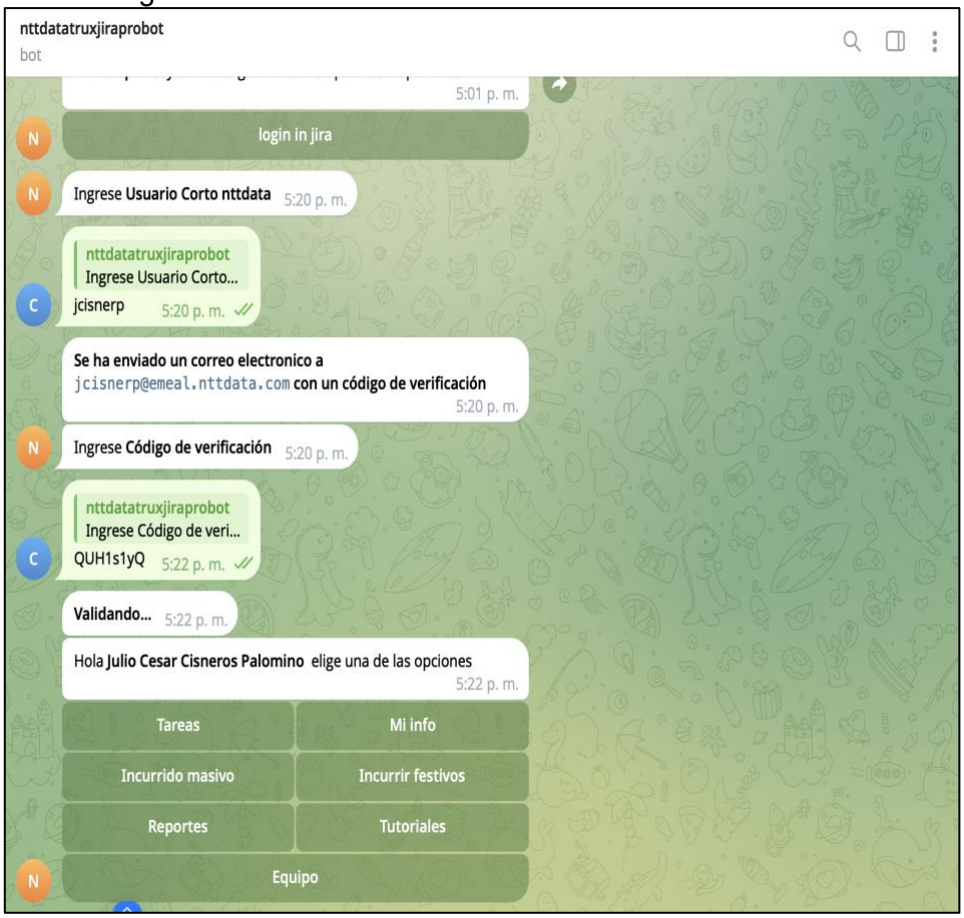


Figura 7
Menú principal bot iohara

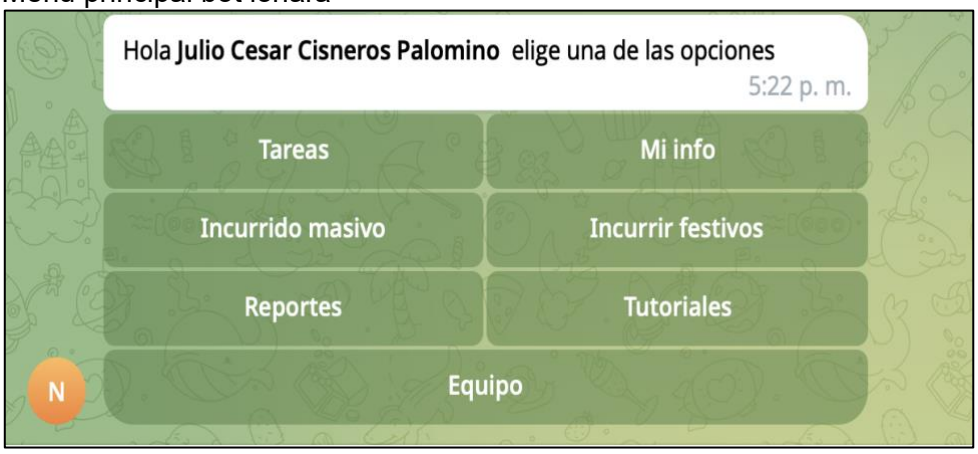


Figura 8

Opción Tareas

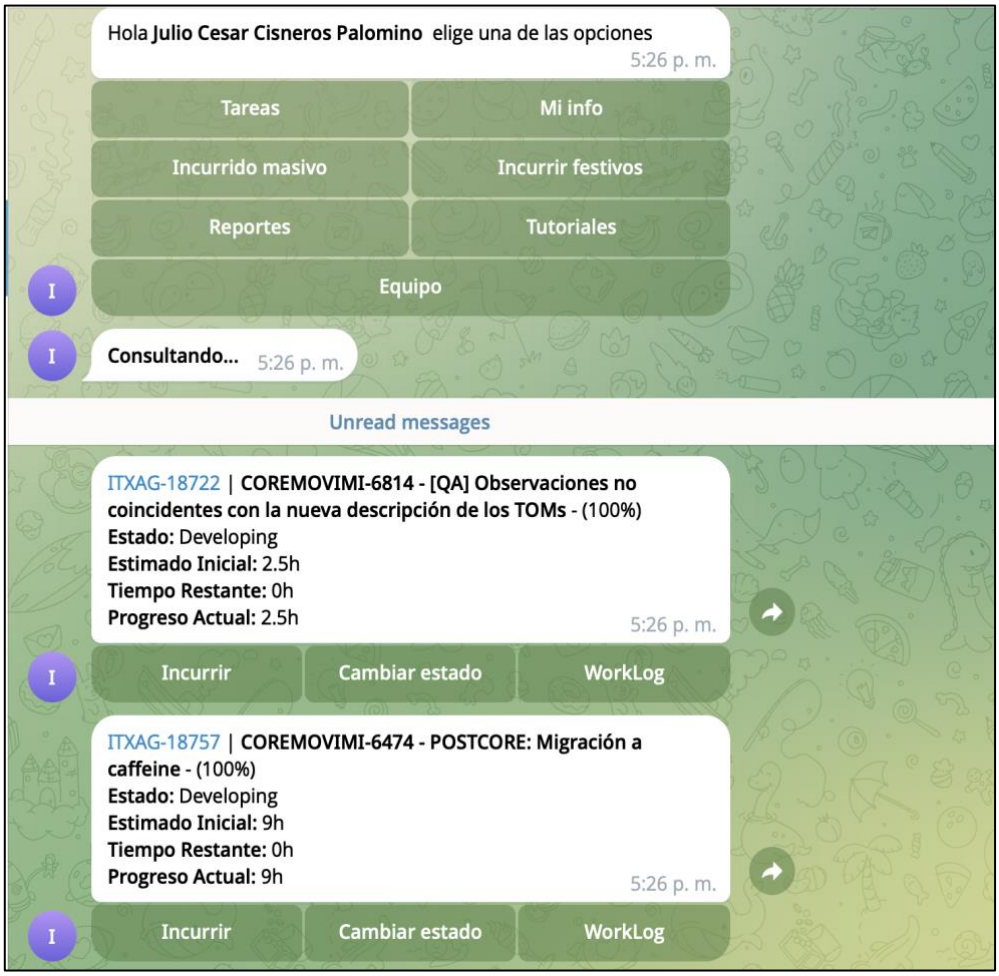


Figura 9

Opción incurrir

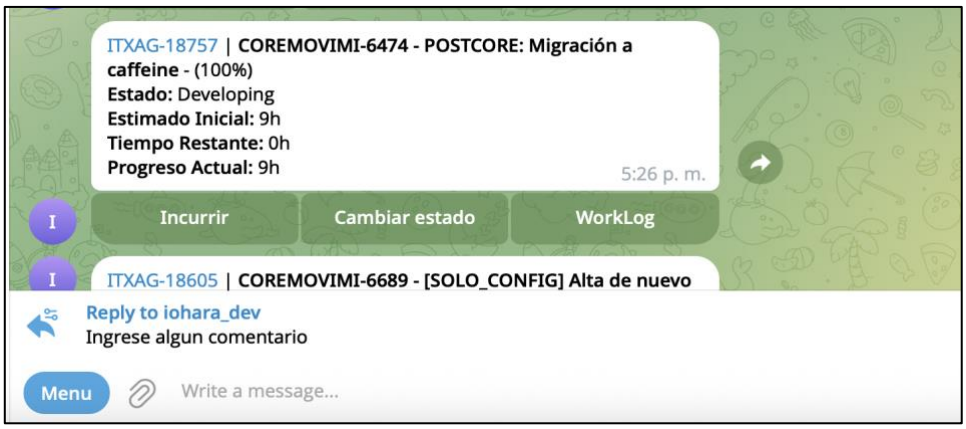


Figura 10

Opción cambiar estado

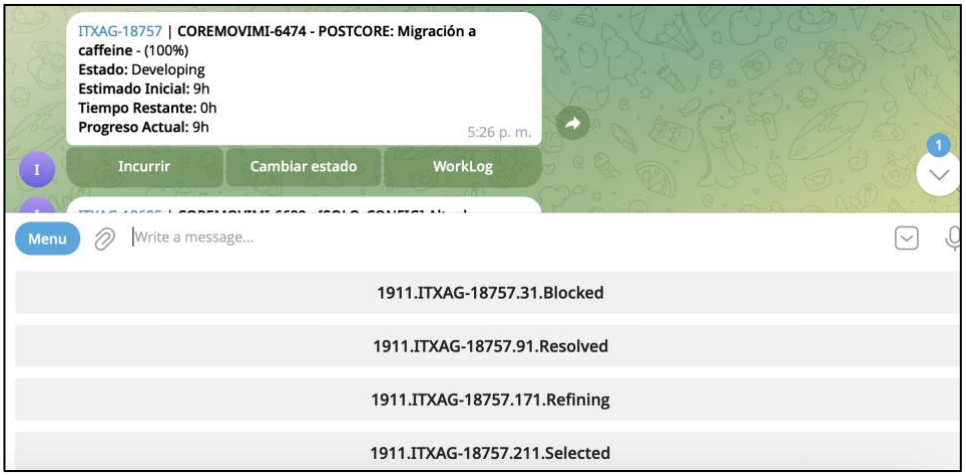


Figura 11

Opción worklog

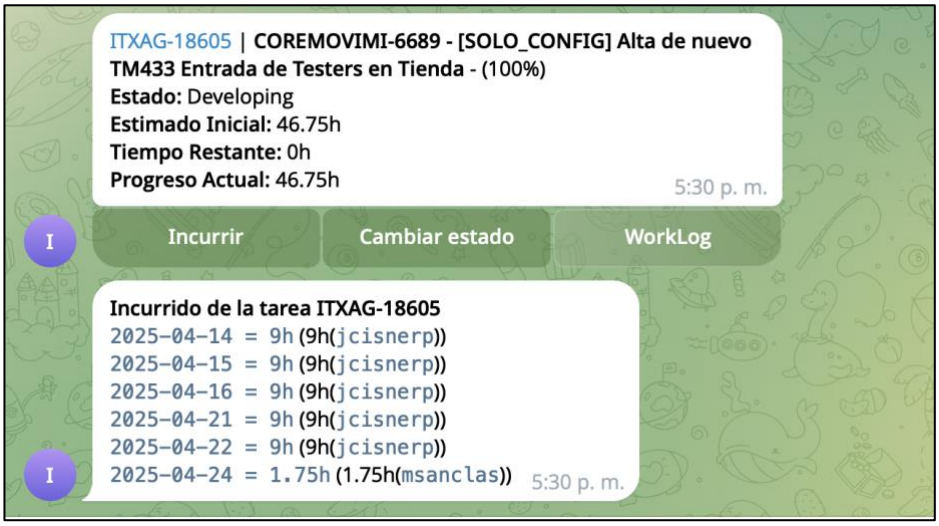


Figura 12

Opción “Mi info”

Figura 13

Opción Incurrido masivo

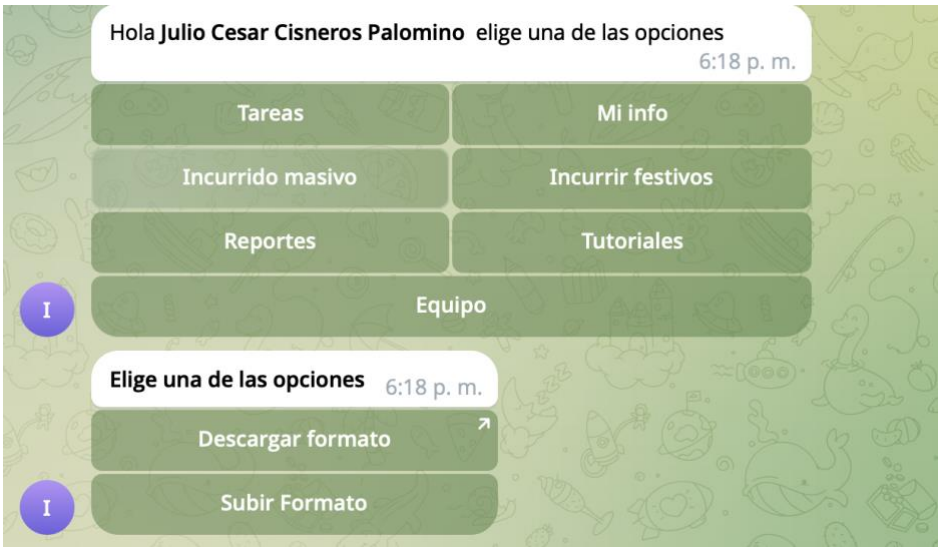


Figura 14

Opción Incurrir festivos

Figura 15

Opción reportes

Figura 16

Opción de incurrido

Figura 17

Opción Reporte Excel

Figura 18

Opción de Excel

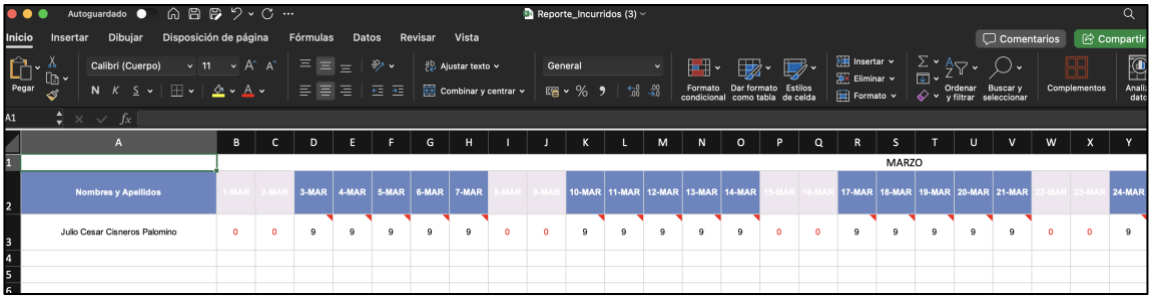


Figura 19

Opción Incurrir comando /worklog

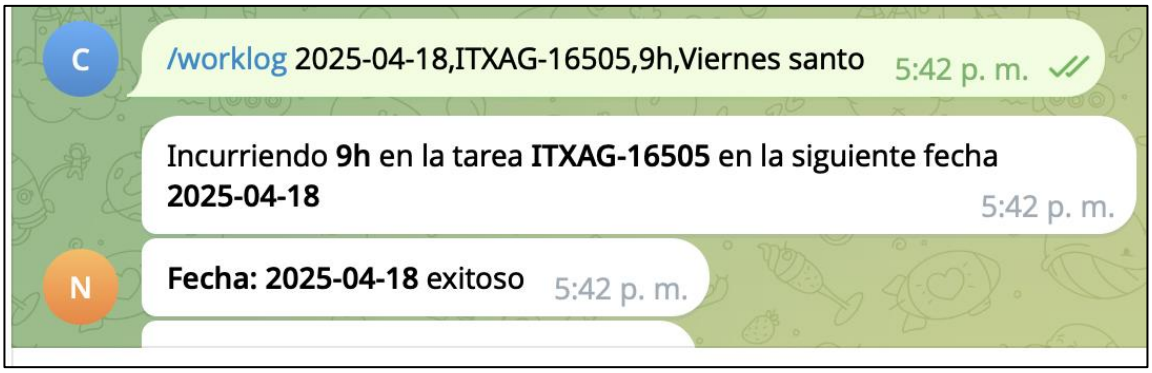
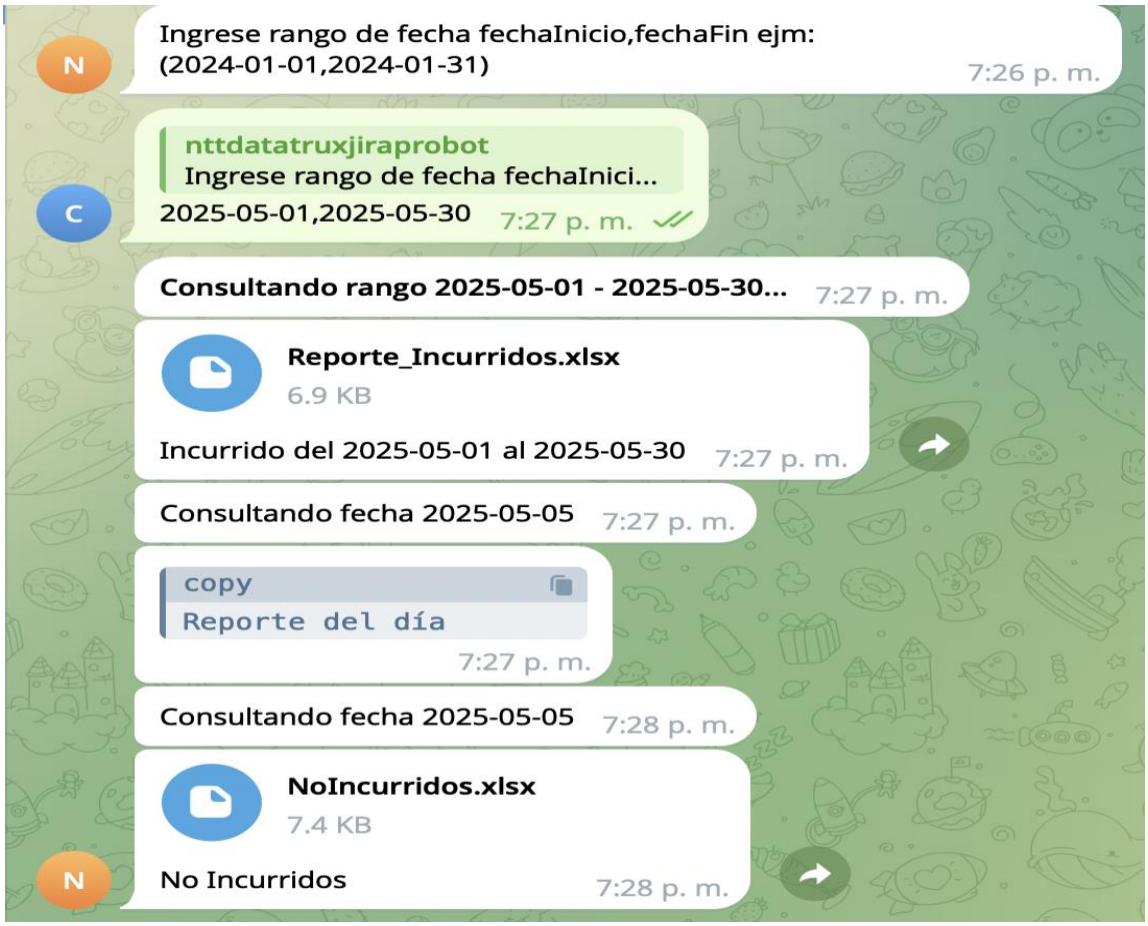


Figura 20

Opción Equipos

Figura 21

Opción de Equipos



TARJETAS CLASE RESPONSABILIDAD COLABORACIÓN (CRC)

Para un diseño simple, continuamos definiendo la tarjeta CRC, que puede identificar las clases, sus responsabilidades y la colaboración realizada para las historias de usuario y las tareas correspondientes.

Clase Festivo

Tabla 49. Lista de clase festivo

CLASE: Festivo	
Responsabilidades: Mantener el registro de festivos para incurrir automaticamente.	**Colaboradores:** Ningumo **Atributos:** Fecha Comentarios Hora Oficina

Clase Equipo

Tabla 50. Lista de clase equipo

CLASE: Equipo	
Responsabilidades: Gestionar informacion de equipos y sus miembros, lideres y managers	**Colaboradores:** Ningumo **Atributos:** Key Peoples EmailsSuper emailsLeader emailsServiceLeader alerTeams jiraServices

Tabla 51. Lista de clase User

CLASE: User	
Responsabilidades: Gestionar usuarios autenticados en el bot y sus roles.	**Colaboradores:** Ningumo **Atributos:** isTelegram displayName token key name emailAddress isFestivo isSupervisor proyectsJira supervisadoPor timeZone notAutomatedIncurred active rol hub

Clase People

Tabla 52. Lista de clase People

CLASE: People	
Responsabilidades: Mantener informacion de empleados y sus asignaciones.	**Colaboradores:** Ningumo **Atributos:** idEmpleado nombres apellidos usuario categoria hub estado fechaAlta fechaBaja tecnologia asignacion

Figura 22
CÓDIGO FUENTE
DAO

```
'use strict'
const MongoDB = require('../databases/mongo') ;
const { ObjectId } = require('mongodb');
async function  get(key,valor, tableName) {
    let result = await MongoDB.getMongoConnection().then(async resp => {
        const data = await resp.db.collection(tableName).findOne({[key]: valor});
        await resp.client.close();
        return data;
    });
    return result;
}

async function  getBy_id(id, tableName) {
    let result = await MongoDB.getMongoConnection().then(async resp => {
        const objectId = new ObjectId(id);
        const data = await resp.db.collection(tableName).findOne({_id: objectId});
        await resp.client.close();
        return data;
    });
    return result;
}


async function  get2(filters, tableName) {
    let result = await MongoDB.getMongoConnection().then(async resp => {
        const data = await resp.db.collection(tableName).findOne(filters);
        await resp.client.close();
```

```
        return data;
    });
    return result;
}

async function list(projection = {}, filters = {}, tableName) {
    let resp = await MongoDB.getMongoConnection().then(async resp => {
        const data = await resp.db.collection(tableName).find(filters).project(projection);
        //await resp.client.close();
        return {data,client:resp.client};
    }).then(async x => {
        const y = await x.data.toArray()
       await x.client.close();
      return y;
    });

    if (!resp) {
        return [];
    }

    return resp;
}

async function save(data, tableName){
    let resp = await MongoDB.getMongoConnection().then(async resp => {
        const resultado = await resp.db.collection(tableName).insert(data);
        await resp.client.close();
        return resultado
    })
    return resp;
}

async function update(filter, setters,tableName) {
    let result = await MongoDB.getMongoConnection().then(async resp => {
        const data = await resp.db.collection(tableName).update(filter, {$set: setters});
        await resp.client.close();
        return data;
    });
    return result;
}

async function saveList(dataList,tableName) {
    let resp = await  MongoDB.getMongoConnection().then(async resp => {
        const bulk = await resp.db.collection(tableName).initializeUnorderedBulkOp();
        await resp.db.collection(tableName).deleteMany({});
        dataList.forEach(todo => {
            bulk.insert(todo);
        });
        return  bulk.execute();
    })
    return resp;
}

module.exports = {
    get,
```

```
    list,
    save,
    update,
    saveList,
    get2,
    getBy_id
}
```

Figura 23
DAO EQUIPOS

```
'use strict'

const MongoDao = require('./base');

const tableName = process.env.EQUIPOS_COLLECTIONS;

async function list(){
    const result = await MongoDao.list({},{},tableName);
    return result;
}
async function obtenerEquipoByUser(usuario) {
    return await MongoDao.list({},{peoples:{$all:[usuario]}}, tableName);
}

async function listByFilters(filters){
    const result = await MongoDao.list({},filters,tableName);
    return result;
}

async function getId(id){
    const result = await MongoDao.getBy_id(id, tableName);
    return result;
}

module.exports = {
    list,
    obtenerEquipoByUser,
    listByFilters,
    getId
}
```

Figura 24
DAO FESTIVOS

```
'use strict'

const MongoDao = require('./base');

const tableName = process.env.FESTIVOS_COLLECTIONS;


async function list(){
    const result = await MongoDao.list({},{},tableName);
    return result;
}
```

```
module.exports = {
   list
}
```

Figura 25
DAO LOGSREPORT

```
'use strict'

const MongoDao = require('./base');

const tableName = process.env.LOGS_REPORTES_COLLECTIONS;

async function list(){
   const result = await MongoDao.list({},{},tableName);
   return result;
}
module.exports = {
   list
}
```

Figura 26
DAO PEOPLES

```
'use strict'

const MongoDao = require('./base');

const tableName = process.env.PEOPLES_COLLECTIONS;

async function saveList(lista) {
   return await MongoDao.saveList(lista, tableName);
}

async function obtenerPersona(usuario) {
   return await MongoDao.get2({$or:[{idEmpleado:usuario},{usuario:usuario}]},
tableName);
}


module.exports = {
   saveList,
   obtenerPersona
}
```

Figura 27
DAO USERS

```
'use strict'

const MongoDao = require('./base');

const tableName = process.env.USERS_COLLECTIONS;

async function saveUsers(data){
```

```
    const result = await MongoDao.save(data, tableName);
    return result;
}

async function getUsers(idTelegram){
    const result = await MongoDao.get('idTelegram',idTelegram, tableName);
    return result;
}

async function getUserByName(name){
    const result = await MongoDao.get('name',name, tableName);
    return result;
}

async function updateUsers(filter,setter){
    const result = await MongoDao.update(filter,setter, tableName);
    return result;
}

async function listAllUsers(){
    const result = await
MongoDao.list({idTelegram:1,name:1,displayName:1,notAutomatedIncurred:1,
token:1},{}, tableName);
    return result;
}

module.exports = {
    saveUsers,
    getUsers,
    updateUsers,
    getUserByName,
    listAllUsers
}
```

Figura 28
SERVICES
CACHE SERVICE

```
'use strict'
const Cache = require('ttl');
const UsersDao = require('../dao/users.dao');

let cache = new Cache({
    ttl: 60 * 60 * 1000,
    capacity: 10
});

async function getCache(key) {
    let cap = cache.get(key);
    let result = null;
    if (cap === undefined || cap === null) {
        result = await UsersDao.getUsers(key);
        cache.put(key, result);
    } else {
        result = cap;
    }
```

```
    return result;
}

async function getCache2(key) {
    let cap = cache.get(key);
    let result = null;
    if (cap !== undefined || cap !== null) {
        result = cap;
    }
    return result;
}

function setCache(key, valor) {
    cache.put(key, valor);
}

function deleteCache(key) {
    cache.del(key);
}

function clearCache() {
    cache.clear();
}

module.exports = {
    getCache,
    setCache,
    deleteCache,
    clearCache,
    getCache2
}
```

Figura 29
SERVICE EMAIL

```
'use strict'
const nodemailer = require("nodemailer");
const user = process.env.EMAIL_USER;
const pass = process.env.EMAIL_PASSWORD;
const email = process.env.EMAIL_MASTER;

async function sendEmailFile(toEmail,subject,
bodyHtml,nameFile,buffer,bot,idTelegram){

    let transporter = nodemailer.createTransport({
        host: "relay.emeal.nttdatareports.com",
        port: 587,
        secure: false,
        auth: {
            user: user,
            pass: pass,
        },
    });
    let info = await transporter.sendMail({
        from: email,
        to: toEmail,
```

```
        subject: subject,
        html: bodyHtml,
        attachments: [{
            filename: `${nameFile}.xlsx`,
            content: new Buffer.from(buffer)
        }]
    });
    if(bot && idTelegram){
        bot.sendMessage(idTelegram, info.messageId);
    }
    console.log("Message sent: %s", info.messageId);
}

async function sendEmailFile2(toEmail,subject, bodyHtml,nameFile,buffer){

    let transporter = nodemailer.createTransport({
        host: "relay.emeal.nttdatareports.com",
        port: 587,
        secure: false,
        auth: {
            user: user,
            pass: pass,
        },
    });
    let info = await transporter.sendMail({
        from: email,
        to: toEmail,
        cc: email,
        subject: subject,
        html: bodyHtml,
        attachments: [{
            filename: `${nameFile}.xlsx`,
            content: new Buffer.from(buffer)
        }]
    });

    console.log("Message sent: %s", info.messageId);
}

async function sendEmail(toEmail,subject, bodyHtml){

    let transporter = nodemailer.createTransport({
        host: "relay.emeal.nttdatareports.com",
        port: 587,
        secure: false,
        auth: {
            user: user,
            pass: pass,
        },
    });
    let info = await transporter.sendMail({
        from: email,
        to: toEmail,
        subject: subject,
        html: bodyHtml
```

```
    });

    console.log("Message sent: %s", info.messageId);
}

async function sendEmailRobot(toEmail,subject, bodyHtml){

    let transporter = nodemailer.createTransport({
        host: "relay.emeal.nttdatareports.com",
        port: 587,
        secure: false,
        auth: {
            user: user,
            pass: pass,
        },
    });
    let info = await transporter.sendMail({
        from: email,
        to: toEmail,
        subject: subject,
        html: bodyHtml
    });

    console.log("Message sent: %s", info.messageId);
}
module.exports = {
    sendEmailFile,
    sendEmailFile2,
    sendEmail,
    sendEmailRobot
}
```

Figura 30
SERVICE EQUIPOS

```
'use strict'
const { getRoleMongo } = require('../constante');
const EquiposDao = require('../dao/equipos.dao');

async function listEquipos(filter, setter){
    const result = await EquiposDao.list();
    return result;
}

async function findEquipoByUser(user){
    const result = await EquiposDao.obtenerEquipoByUser(user);
    return result;
}

async function findEquipoByRole(role, user){
    const result = await EquiposDao.listByFilters({[getRoleMongo(role)]:user});
    let chunkSize = 2;
    let dividedArrays = [[{
        text: `All`,
        callback_data: JSON.stringify({
            idequipo: 'allE'
```

```
        })
    }]];
    let resultado = [];
    for(let eq of result){
        resultado.push({
            text: `${eq.key}`,
            callback_data: JSON.stringify({
                idequipo: eq._id
            })
        });
    }
    for (let i = 0; i < resultado.length; i += chunkSize) {
        let chunk = resultado.slice(i, i + chunkSize);
        dividedArrays.push(chunk);
    }
    return dividedArrays;
}

async function findEquipoByRoleV2(role, user){
    const result = await EquiposDao.listByFilters({[getRoleMongo(role)]:user});

    let resultado = [{mensaje:'<b>Service Team: </b> ALL TEAM' + '\n',menu:[[{
        text: `Reporte Rango`,
        callback_data: JSON.stringify({
            idequipo: 'allE'
        })
    }],[{
        text: `Reporte SMS`,
        callback_data: JSON.stringify({
            teamSms: 'allE'
        })
    }],[{
        text: `Alerta No Incurrido`,
        callback_data: JSON.stringify({
            teamNoI: 'allE'
        })
    }]]}];
    for(let eq of result){

        const m = '<b>Service Team: </b>' + `${eq.key}` + '\n'

        const botones =  [[{
            text: `Reporte Rango`,
            callback_data: JSON.stringify({
                idequipo: eq._id
            })
        },{
            text: `Reporte SMS`,
            callback_data: JSON.stringify({
                teamSms: eq._id
            })
        }],[{
            text: `Alerta No Incurrido`,
            callback_data: JSON.stringify({
                teamNoI: eq._id
```

```
            })
        }]];

        const final = {mensaje: m, menu: botones};
        resultado.push(final)

    }

    return resultado;
}

async function peoplesByUserRoleIdEquipo(role, user, id){
    let peoples = [];
    if(id === 'allE'){
     const   result  = await EquiposDao.listByFilters({[getRoleMongo(role)]:user});
        for(let eq of result){
            peoples = peoples.concat(eq.peoples);
        }
    }else{
        const result = await EquiposDao.getId(id);
        peoples =  peoples.concat(result.peoples);
    }


    return peoples;
}


module.exports = {
    listEquipos,
    findEquipoByUser,
    findEquipoByRole,
    peoplesByUserRoleIdEquipo,
    findEquipoByRoleV2
}
```

Figura 31
SERVICE EXCEL

```
const ExcelJS = require("exceljs");
const UtilsService = require("./utils.service");
const PeopleService = require("./people.service");

async function reportePrueba(dataSet, mes, fechas) {
  const workbook = new ExcelJS.Workbook();
  const worksheet = workbook.addWorksheet("Reporte Incurrido");

  const cabeceras = fechas.map((x) => x.fecha.substring(8, 10));
  const sabDom = fechas
    .filter((x) => x.day === "S" || x.day === "D")
    .map((x) => x.fecha.substring(8, 10));

  const columnas = [];
  const filaMes = worksheet.addRow(["", mes.toUpperCase()]);
  filaMes.alignment = { horizontal: "center", vertical: "middle" };
```

```
const header = ["Nombres y Apellidos", ...cabeceras];
const headerRow = worksheet.addRow(header);
headerRow.alignment = { horizontal: "center", vertical: "middle" };
headerRow.height = 30;
headerRow.eachCell((cell, num) => {
  if (sabDom.includes(cell.value)) {
    cell.fill = {
      pattern: "solid",
      type: "pattern",
      fgColor: { argb: "F0E6EF" },
    };
  } else {
    cell.fill = {
      pattern: "solid",
      type: "pattern",
      fgColor: { argb: "6785C1" },
    };
  }
  cell.border = {
    top: { style: "thin", color: { argb: "D9D9D9" } },
    left: { style: "thin", color: { argb: "D9D9D9" } },
    bottom: { style: "thin", color: { argb: "D9D9D9" } },
    right: { style: "thin", color: { argb: "D9D9D9" } },
  };
  cell.font = {
    name: "Arial",
    family: 4,
    size: 12,
    color: { argb: "ffffff" },
    bold: true,
  };
  columnas.push(cell.address);
});
worksheet.mergeCells(`B1:${columnas[columnas.length - 1].replace("2", "1")}`);
const filadata = worksheet.addRow(dataSet);
filadata.height = 25;
filadata.alignment = {
  vertical: "middle",
  horizontal: "center",
  wrapText: true,
};
/* dataSet.forEach((d, num) => {
          const row = worksheet.addRow(d);
          row.height = 25;
          row.alignment = {vertical: 'middle', horizontal: 'center', wrapText: true};
      }
   );*/
worksheet.getColumn(1).width = 30;
for (let e = 2; e < 500; e++) {
  worksheet.getColumn(e).width = 5;
}

worksheet.addRow([]);
return await workbook.xlsx.writeBuffer();
```

```
}

async function reportePorUsuario(dataSet, mes, fechas) {
  const HRS_REGISTRADAS = "Hrs Registradas";
  const HRS_ESPERADAS = "Hrs Esperadas";
  const HRS_FALTA = "Falta Registrar";
  const workbook = new ExcelJS.Workbook();
  const worksheet = workbook.addWorksheet("Reporte Incurrido");

  const cabeceras = fechas.map((x) =>
    UtilsService.convertFechaReporte(x.fecha)
  );
  const sabDom = fechas
    .filter((x) => x.day === "S" || x.day === "D")
    .map((x) => UtilsService.convertFechaReporte(x.fecha));
  const adicional = [HRS_REGISTRADAS, HRS_ESPERADAS, HRS_FALTA];

  const columnas = [];
  let columnasSabDom = [];
  let columnasLuVi = [];
  const filaMes = worksheet.addRow(["", mes.toUpperCase()]);
  filaMes.alignment = { horizontal: "center", vertical: "middle" };

  const header = ["Nombres y Apellidos", ...cabeceras, ...adicional];
  const headerRow = worksheet.addRow(header);
  headerRow.alignment = {
    horizontal: "center",
    vertical: "middle",
    wrapText: true,
  };
  headerRow.height = 30;
  let codHrsRegistradas = "";
  let codHrsEsperadas = "";
  let codHrsFalta = "";
  headerRow.eachCell((cell, num) => {
    if (sabDom.includes(cell.value)) {
      cell.fill = {
        pattern: "solid",
        type: "pattern",
        fgColor: { argb: "F0E6EF" },
      };
      columnasSabDom.push(cell.address);
    } else {
      cell.fill = {
        pattern: "solid",
        type: "pattern",
        fgColor: { argb: "6785C1" },
      };
      if (!adicional.includes(cell.value)) {
        columnasLuVi.push(cell.address);
      }
    }
    if (cell.value === HRS_REGISTRADAS) {
      codHrsRegistradas = UtilsService.eliminarUltimaLetra(cell.address);
    }
```

```
  if (cell.value === HRS_ESPERADAS) {
    codHrsEsperadas = UtilsService.eliminarUltimaLetra(cell.address);
  }
  if (cell.value === HRS_FALTA) {
    codHrsFalta = UtilsService.eliminarUltimaLetra(cell.address);
  }
  cell.border = {
    top: { style: "thin", color: { argb: "D9D9D9" } },
    left: { style: "thin", color: { argb: "D9D9D9" } },
    bottom: { style: "thin", color: { argb: "D9D9D9" } },
    right: { style: "thin", color: { argb: "D9D9D9" } },
  };
  cell.font = {
    name: "Arial",
    family: 4,
    size: 8,
    color: { argb: "ffffff" },
    bold: true,
  };
  columnas.push(cell.address);
});

worksheet.mergeCells(`B1:${columnas[columnas.length - 1].replace("2", "1")}`);
let filas = [];
dataSet.forEach((d, num) => {
  const horas = d.map((x, y) =>
    y !== 0 ? Number(x.split("|")[0]) : x.split("|")[0]
  );
  const comentario = d.map((x, y) =>
    x.split("|")[1] ? x.split("|")[1] : ""
  );
  const row = worksheet.addRow(horas);
  row.height = 25;
  row.alignment = {
    vertical: "middle",
    horizontal: "center",
    wrapText: true,
  };
  row.font = { name: "Arial", family: 4, size: 8 };
  filas.push(row.number);
  row.eachCell((cell, num) => {
    if (cell.value !== 0 && num !== 1) {
      cell.note = {
        texts: [
          {
            font: {
              size: 10,
              color: { theme: 1 },
              name: "Arial",
              family: 2,
              scheme: "minor",
            },
            text: comentario[num - 1],
          },
        ],
```

```
        margins: {
          insetmode: "custom",
          inset: [0.25, 0.25, 0.35, 0.35],
        },
        protection: {
          locked: true,
          lockText: true,
        },
      };
    }
  });
});
filas.forEach((f) => {
  worksheet.getCell(codHrsRegistradas + f).value = {
    formula: columnasLuVi
      .filter((col, i) => i !== 0)
      .map((a) => UtilsService.eliminarUltimaLetra(a) + f)
      .join("+"),
  };
  worksheet.getCell(codHrsEsperadas + f).value =
    columnasLuVi.filter((col, i) => i !== 0).length * 9;
  worksheet.getCell(codHrsFalta + f).value = {
    formula: `${codHrsRegistradas + f}-${codHrsEsperadas + f}`,
  };
  if (worksheet.getCell(codHrsFalta + f).value < 0) {
    worksheet.getCell(codHrsFalta + f).font = {
      name: "Arial",
      family: 4,
      size: 8,
      color: { argb: "ff0000" },
    };
  }
  worksheet.getRow(f).eachCell((cell, num) => {
    if (num !== 0) {
      if (cell.value < 9) {
        cell.font = {
          name: "Arial",
          family: 4,
          size: 8,
          color: { argb: "ff0000" },
        };
      }
    }
  });
});
worksheet.getColumn(1).width = 30;
for (let e = 2; e <= cabeceras.length + 1; e++) {
  worksheet.getColumn(e).width = 6;
}

worksheet.addRow([]);
//  worksheet.addRow(['Leyenda:']);
worksheet.addRow([]);
return await workbook.xlsx.writeBuffer();
}
```

```
async function leerExcel(buffer, bot, id) {
  const wb = new ExcelJS.Workbook();
  let personas = [];
  const asignacion = [];
  wb.xlsx.load(buffer).then(async (workbook) => {
    workbook.eachSheet((sheet, id) => {
      if (sheet.orderNo === 0) {
        sheet.eachRow((row, rowIndex) => {
          if (rowIndex !== 1) {
            const data = mapperPeople(row.values);
            personas.push(data);
          }
        });
      } else if (sheet.orderNo === 2) {
        sheet.eachRow((row, rowIndex) => {
          if (rowIndex !== 1) {
            const data = mapperPeopleAsignacion(row.values);
            asignacion.push(data);
          }
        });
      }
    });
    for (let persona of personas) {
      const asig = asignacion.filter(
        (a) => a.idEmpleado === persona.idEmpleado
      );
      persona.asignacion = asig;
    }
    await PeopleService.savePeoples(personas);
    bot.sendMessage(id, ` <b>CARGA TERMINADA</b>`, {
      parse_mode: "HTML",
    });
    console.log("CARGA TERMINADA");
  });
}

async function leerExcel2(buffer) {
  const wb = new ExcelJS.Workbook();
  let filas = [];
  const promise = new Promise(async function (resolve, reject) {
    wb.xlsx.load(buffer).then(async (workbook) => {
      workbook.eachSheet((sheet, id) => {
        if (sheet.orderNo === 0) {
          sheet.eachRow((row, rowIndex) => {
            filas.push({ row: row.values, rowIndex });
          });
        }
      });
      resolve(filas);
    });
  });
  return promise;
}
```

```
async function leerExcelEliminar(buffer) {
  const wb = new ExcelJS.Workbook();
  let dataEliminar = [];
  const promise = new Promise(async function (resolve, reject) {
    wb.xlsx.load(buffer).then(async (workbook) => {
      workbook.eachSheet((sheet, id) => {
        if (sheet.orderNo === 0) {
          sheet.eachRow((row, rowIndex) => {
            if (rowIndex !== 0 && rowIndex !== 1 && rowIndex !== 2) {
              const data = mapperWorklogEliminar(row.values);
              dataEliminar.push(data);
            }

          });
        }
      });
      resolve(dataEliminar);
    });
  });
  return promise;
}

async function reporteNoIncurridos(dataSet) {
  const workbook = new ExcelJS.Workbook();
  const worksheet = workbook.addWorksheet("Reporte Incurrido");

  worksheet.addTable({
    name: "NoIncurridos",
    ref: "A1",
    headerRow: true,
    style: {
      theme: "TableStyleLight1",
      showRowStripes: true,
    },
    columns: [
      { name: "usuario" },
      { name: "correo" },
      { name: "fecha" },
      { name: "hora" },
      { name: "mensaje" },
    ],
    rows: dataSet,
  });
  return await workbook.xlsx.writeBuffer();
}

async function reporteGrupoTeams(dataSet) {
  const workbook = new ExcelJS.Workbook();
  const worksheet = workbook.addWorksheet("Data Teams");

  worksheet.addTable({
    name: "DataTeams",
    ref: "A1",
    headerRow: true,
    style: {
```

```
      theme: "TableStyleLight1",
      showRowStripes: true,
    },
    columns: [
      { name: "equipo" },
      { name: "grupoTeams" },
      { name: "path" }
    ],
    rows: dataSet,
  });
  return await workbook.xlsx.writeBuffer();
}

function mapperPeople(data) {
  return {
    idEmpleado: "" + data[1],
    nombres: data[2]?.toUpperCase(),
    apellidos: data[3]?.toUpperCase(),
    usuario: data[4],
    categoria: data[5]?.toUpperCase(),
    hub:data[6]?.includes('Trujillo') ? 'TRU': data[6]?.includes('Arequipa') ? 'ARE': 'LIM',
    estado: data[9]?.toUpperCase(),
    fechaAlta: data[10],
    fechaBaja: data[11],
    tecnologia: data[12]?.toUpperCase(),
  };
}

function mapperPeopleAsignacion(data) {
  return {
    idEmpleado: "" + data[1],
    servicio: data[5]?.toUpperCase(),
    servicioTeam: data[7]?.toUpperCase(),
    asignacion: data[10],
    fechaInicio: data[11]?.toUpperCase(),
    fechaFin: data[12]?.toUpperCase(),
    tecnologia: data[14]?.toUpperCase(),
  };
}

function mapperWorklogEliminar(data) {
  return {
    id:  data[1],
    jirakey: data[2]?.toUpperCase(),
    fecha: data[3]?.toUpperCase(),
    hora: data[4],
    comentario: data[5]?.toUpperCase(),
    eliminar: data[6]?.toUpperCase()
  };
}


async function reporteEliminar(dataSet, usuario) {
  const workbook = new ExcelJS.Workbook();
  const worksheet = workbook.addWorksheet(`${usuario}`);
```

```
 const columnas = [];
 const filaMes = worksheet.addRow(["* Seleccione los worklogs a eliminar en la columna ELIMINAR con un SI"]);
 filaMes.alignment = { horizontal: "center", vertical: "middle" };
 filaMes.color={ fgColor: { argb: "6785C1" },}

 const header = ["ID", "Issue","Fecha","Hora","Comentario","Eliminar"];
 const headerRow = worksheet.addRow(header);
 headerRow.alignment = { horizontal: "center", vertical: "middle" };
 headerRow.height = 30;
 worksheet.getCell('A1').font = {
  name: "Arial",
  family: 4,
  size: 12,
  color: { argb: "ff0000" },
  bold: true,
 };
 headerRow.eachCell((cell, num) => {
  cell.fill = {
   pattern: "solid",
   type: "pattern",
   fgColor: { argb: "6785C1" },
  };
  cell.border = {
   top: { style: "thin", color: { argb: "D9D9D9" } },
   left: { style: "thin", color: { argb: "D9D9D9" } },
   bottom: { style: "thin", color: { argb: "D9D9D9" } },
   right: { style: "thin", color: { argb: "D9D9D9" } },
  };
  cell.font = {
   name: "Arial",
   family: 4,
   size: 12,
   color: { argb: "ffffff" },
   bold: true,
  };
  columnas.push(cell.address);
 });
 worksheet.mergeCells(`A1:${columnas[columnas.length - 1].replace("2", "1")}`);

dataSet.forEach((d, num) => {
        const row = worksheet.addRow(d);
        row.height = 25;
        row.alignment = {vertical: 'middle', horizontal: 'center', wrapText: true};
        row.eachCell((cell,num)=>{
         if(num === 6){
           cell.dataValidation = {
            type: 'list',
            allowBlank: false,
            formulae: ['"NO,SI"']
           }
         }

        })
```

```
        }
    );

  worksheet.getColumn(1).width = 30;
  for (let e = 2; e < 500; e++) {
    worksheet.getColumn(e).width = 30;
  }

  worksheet.addRow([]);
  return await workbook.xlsx.writeBuffer();
}

module.exports = {
  reportePrueba,
  reportePorUsuario,
  leerExcel,
  reporteNoIncurridos,
  leerExcel2,
  reporteEliminar,
  leerExcelEliminar,
  reporteGrupoTeams
};
```

Figura 32
SERVICE JIRA

```
'use strict'

const axios = require('axios');
const dateFormat = require('dateformat');
const CacheService = require('./cache.service');
const UsersDao = require('../dao/users.dao');
const PeopleDao = require('../dao/peoples.dao');

const fs = require('fs');
const UtilsService = require('./utils.service');
const constantes = require('../constante');
const { log } = require('console');

const baseUrl = process.env.BASE_URL;
const baseUrl2 = process.env.BASE_URL2;
const tokenRobot = process.env.TOKEN_ROBOT;

async function addWorklog(text, id, key, comentario, fechaI) {

    const data = await CacheService.getCache(id);
    const timezone = data.timeZone ? data.timeZone : "America/Lima"
    var now = null;
    if (fechaI == null) {
        now = new Date().toLocaleString('en-US', {
            timeZone: timezone
        });

    } else {
        now = new Date(fechaI+'T13:00:00.000+0100').toLocaleString('en-US', {
            timeZone: timezone
```

```
        });
    }
    const fecha = dateFormat(now, "yyyy-mm-dd'T'HH:MM:ss.000+0100");
    const hora = UtilsService.converte(text);
    return Promise.resolve('worlog')
        .then(result => {
            const config1 = {
                headers: {Authorization: `Basic ${data.token}`}
            };
            let urlUserJira = `${baseUrl}/issue/${key}/worklog`;
            const body = {timeSpentSeconds: hora, started: fecha, comment: comentario};
            return axios.post(`${urlUserJira}`, body, config1).then(resp => {
                const response = resp.data.timeSpent;

                return response;
            }, error => {

                throw Error('time incorrect');
            });

        });
}

async function addWorklog2(text, id, key, comentario, fechaI,usuario) {

    const data = await CacheService.getCache(id);
    const timezone = data.timeZone ? data.timeZone : "America/Lima"
    const user = usuario && UtilsService.verifircarPermisoAdmin(data.rol) ? usuario :
data.name;
    var now = null;
    if (fechaI == null) {
         now = new Date().toLocaleString('en-US', {
            timeZone: timezone
        });

    } else {
        now = new Date(fechaI+'T13:00:00.000+0100').toLocaleString('en-US', {
            timeZone: timezone
        });
    }
    const fecha = dateFormat(now, "dd/mm/yyyy HH:MM");
    const hora = UtilsService.converte(text);
    return Promise.resolve('worlog')
        .then(result => {
            const config1 = {
                headers: {Authorization: `Basic ${tokenRobot}`}
            };
            let urlUserJira = `${baseUrl2}/scriptrunner/latest/custom/generateWorkLog`;
            const body = {issueKey: key, userName: user, comment:
comentario,timeSpent:hora,dateLog:fecha};
            return axios.put(`${urlUserJira}`, [body], config1).then(resp => {
                const response = resp.data;

                return response;
            }, error => {
```

```
            throw Error('time incorrect');
          });

      });
}

async function deleteWorklog(id,jiraKey,worklogId) {
   const data = await CacheService.getCache(id);
   return Promise.resolve('worlog')
   .then(result => {
      const config1 = {
         headers: {Authorization: `Basic ${tokenRobot}`}
      };
      let urlUserJira = `${baseUrl}/issue/${jiraKey}/worklog/${worklogId}`;
      return axios.delete(`${urlUserJira}`, config1).then(resp => {
         return resp;
      }, error => {
         throw Error('time incorrect');
      });

   });
 }

async function getUser(text, id) {
   return Promise.resolve('user')
      .then(result => {

         const config1 = {
            headers: {Authorization: `Basic ${tokenRobot}`}
         };
         let urlUserJira = `${baseUrl}/user?username=${text}`;
         let token = '';
         return axios.get(`${urlUserJira}`, config1).then(async resp => {
            const data = resp.data;
            const usarioMongo = await UsersDao.getUserByName(data.name);
            let people = {hub:'TRU'};
            if(usarioMongo === null || (usarioMongo !== null && !usarioMongo.hub)){
               people = await PeopleDao.obtenerPersona(data.name.toUpperCase());
            }

            const userJira = {
               idTelegram: id,
               displayName: data.displayName,
               key: data.key,
               name: data.name,
               emailAddress: data.emailAddress,
               timeZone:data.timeZone,
               active:false,
               rol:'user',
               hub: people?.hub
            };

            if(usarioMongo){
```

```
                token = usarioMongo.token !== undefined && usarioMongo.token.length
> 8 ? UtilsService.generateClave() : usarioMongo.token;
                userJira.active = usarioMongo.active !== undefined ? usarioMongo.active
: true;
                userJira.rol = usarioMongo.rol !== undefined ? usarioMongo.rol : 'user';
                usarioMongo.token = token;
                await
UsersDao.updateUsers({name:data.name},{...usarioMongo,...userJira});
            }else{
                token = UtilsService.generateClave();
                await UsersDao.saveUsers({...userJira, token ,active:false});
            }

            CacheService.setCache(`${id}`, userJira, 3600 * 1000);
            return {...userJira,token};
        }, error => {
            throw Error('user or token invalid');
        });
    });
}

async function getIssues(text, id) {
   const data = await CacheService.getCache(id);
   return Promise.resolve('issues')
      .then(result => {
         const config1 = {
            headers: {Authorization: `Basic ${tokenRobot}`}
         };
         let urlUserJira = `${baseUrl}/search?jql= assignee=${data.name} and status in
(10409,3,10400,12403)`;
         return axios.get(`${urlUserJira}`, config1).then(resp => {
            const response = resp.data.issues;

            let resultado = [];
            for (let re of response) {
               let botones = [];
               if (re.fields.issuetype.id !== '10200') {//omitimos las epicas
                  var nombreIssue = re.fields.summary;
                  var m = `<a
href="https://umane.emeal.nttdata.com/jiraito/browse/${re.key}">${re.key}</a> |
<b>${nombreIssue}</b> - (${re.fields.progress.percent ? re.fields.progress.percent
:0}%)` + '\n'
                  m += '<b>Estado: </b>' + `${re.fields.status.name}` + '\n'
                  m += '<b>Estimado Inicial: </b>' +
UtilsService.vonvertirHoras(re.fields.progress.total) + '\n'
                  m += '<b>Tiempo Restante: </b>' +
UtilsService.vonvertirHoras(re.fields.progress.total - re.fields.progress.progress) + '\n'
                  m += '<b>Progreso Actual: </b>' +
UtilsService.vonvertirHoras(re.fields.progress.progress) + '\n'

                  const d2 = [{
                     text: `Incurrir`,
                     callback_data: JSON.stringify({
                        command: re.key
                     })
```

```
                    },
                    {
                        text: `Cambiar estado`,
                        callback_data: JSON.stringify({
                            command: 'cambiarE',
                            issueKey: re.key
                        })
                    },

                        {
                            text: `WorkLog`,
                            callback_data: JSON.stringify({
                                command: 'wlIssue',
                                issueKey: re.key
                            })
                        }
                    ]

                    botones.push(d2);
                    const final = {mensaje: m, menu: botones};
                    resultado.push(final)
                }

            }
            return resultado;
        }, error => {
            throw Error('user or token invalid');
        });

    });
}

async function getProyect(text, id) {
    const data = await CacheService.getCache(id);
    return Promise.resolve('proyects')
        .then(result => {
            const config1 = {
                headers: {Authorization: `Basic ${data.token}`}
            };
            let urlUserJira = `${baseUrl}/project`;
            return axios.get(`${urlUserJira}`, config1).then(resp => {
                const response = resp.data;
                let proyectos = [];
                for (let re of response) {

                    const d2 = {key: re.key, name: re.name}
                    proyectos.push(d2);


                }
                return proyectos;
            }, error => {
                throw Error('user or token invalid');
            });
```

```
        });
}

async function getStatusForIssue(issueKey, id) {
    const data = await CacheService.getCache(id);
    return Promise.resolve('updateStatus')
        .then(result => {
            const config1 = {
                headers: {Authorization: `Basic ${tokenRobot}`}
            };
            let urlUserJira = `${baseUrl}/issue/${issueKey}/transitions`;
            return axios.get(`${urlUserJira}`, config1).then(resp => {
                var estados = [];
                for (var res of resp.data.transitions) {
                    if (!res.name.includes("Free")) {
                        var status = constantes.CONDICION_UPDATE_STATUS + '.' +
issueKey + '.' + res.id + '.' + res.to.name;
                        estados.push([status])
                    }

                }
                return estados;
            }, error => {
                throw Error('user or token invalid');
            });

        });
}

async function updateStatus(issueKey, idStatus, id) {
    const data = await CacheService.getCache(id);
    return Promise.resolve('updateStatus')
        .then(result => {
            const config1 = {
                headers: {Authorization: `Basic ${tokenRobot}`}
            };
            let urlUserJira = `${baseUrl}/issue/${issueKey}/transitions`;
            return axios.post(`${urlUserJira}`, {transition: {id: idStatus}}, config1).then(resp
=> {
                return true;
            }, error => {
                throw Error('user or token invalid');
            });

        });
}

async function getWorklog(issueKey, id, filtro, mapaw, user) {
    const data = await CacheService.getCache(id);
    const usuario = user ? user : data.name;
    return Promise.resolve('workLogs')
        .then(result => {
            const config1 = {
                headers: {Authorization: `Basic ${tokenRobot}`}
            };
```

```
            let urlUserJira = `${baseUrl}/issue/${issueKey}/worklog`;
            return axios.get(`${urlUserJira}`, config1).then(async resp => {
                const response = resp.data;
                const logs = response.worklogs;
                var worklogs = filtro ? logs.filter(wl => wl.author.name === usuario) : logs;
                return await proccessDataWorklog(mapaw,worklogs, filtro,issueKey);
            }, error => {
                throw Error('user or token invalid');
            });

        });
}

async function proccessDataWorklog(mapaw,worklogs,filtro,issueKey){
    const promise = new Promise(async function (resolve, reject) {
        var myMap;
        if (mapaw) {
            myMap = mapaw;
        }else{
            myMap = new Map();
        }

        for (let re of worklogs) {
            var mensaje = {sms: '',sms2:'', time: 0,user:'',displayName:''};
            var fecha = new Date(re.started);
            fecha                                                                                    =
UtilsService.converterFechaTimezone3(re.started,re.updateAuthor.timeZone);
            const dataP = `(<code>${filtro ? issueKey : re.author.name}</code>)`;
            const dataP2 = `(${filtro ? issueKey : re.author.name})`;
            if (myMap.has(fecha)) {
                mensaje = myMap.get(fecha);
                mensaje.sms += ", " + UtilsService.vonvertirHoras(re.timeSpentSeconds) +
dataP;
                mensaje.sms2 += ", " + UtilsService.vonvertirHoras(re.timeSpentSeconds)
+ dataP2;
                mensaje.time += re.timeSpentSeconds;
                mensaje.user = re.author.name
                mensaje.displayName = re.author.displayName
                myMap.set(fecha, mensaje);
            } else {
                mensaje.sms = UtilsService.vonvertirHoras(re.timeSpentSeconds) + dataP;
                mensaje.sms2    =    UtilsService.vonvertirHoras(re.timeSpentSeconds)    +
dataP2;
                mensaje.time = re.timeSpentSeconds;
                mensaje.user = re.author.name;
                mensaje.displayName = re.author.displayName;
                myMap.set(fecha, mensaje);
            }

        }
       resolve(myMap);
    })
    return promise;

}
```

```
async function getWorklog2(issueKey, id, startAt,startedAfter, startedBefore) {
   const data = await CacheService.getCache(id);
   return Promise.resolve('workLogs2')
      .then(result => {
         const config1 = {
            headers: {Authorization: `Basic ${tokenRobot}`}
         };
         let urlUserJira =
`${baseUrl}/issue/${issueKey}/worklog?maxResults=5000&startAt=${startAt}&started
After=${startedAfter}&startedBefore=${startedBefore}`;
         return axios.get(`${urlUserJira}`, config1).then(resp => {
            return resp.data;

         }, error => {
            throw Error('user or token invalid');
         });

      });
}


async function getIssuesWorkLog(id, min, max) {
   const data = await CacheService.getCache(id);
   return Promise.resolve('issuesWorklog')
      .then(result => {
         const config1 = {
            headers: {Authorization: `Basic ${tokenRobot}`}
         };
         let urlUserJira = `${baseUrl}/search?jql=worklogAuthor=${data.name} and
worklogDate >= ${min} and worklogDate <= ${max}`;
         return axios.get(`${urlUserJira}`, config1).then(async resp => {
            const response = resp.data.issues;
            let resultado = [];
            var worklogsUser = new Map();
            for (let re of response) {
               let botones = [];
               if (re.fields.issuetype.id !== '10200') {//omitimos las epicas
                  var nombreIssue = re.fields.summary;
                  const final = {id: re.id, key: re.key, name: re.fields.summary};
                  resultado.push(final)
               }
            }
            for (var resul of resultado) {
               await getWorklog(resul.key, id, true,worklogsUser).then(responseW => {
                  UtilsService.addAllMap(worklogsUser, responseW);
               });
            }
            return worklogsUser;
         }, error => {
            throw Error('user or token invalid');
         });

      });
```

```
}

async function getIssuesWorkLogForUser(id, min, max, user) {
   const data = await CacheService.getCache(id);
   const usuario = user ? user : data.name;
   return Promise.resolve('issuesWorklog')
      .then(result => {
         const config1 = {
            headers: {Authorization: `Basic ${tokenRobot}`}
         };
         let urlUserJira = `${baseUrl}/search?jql=worklogAuthor=${usuario} and
worklogDate >= ${min} and worklogDate <= ${max}`;
         return axios.get(`${urlUserJira}`, config1).then(async resp => {
            const response = resp.data.issues;
            let resultado = [];
            var worklogsUser = new Map();
            for (let re of response) {
               let botones = [];
               if (re.fields.issuetype.id !== '10200') {//omitimos las epicas
                  var nombreIssue = re.fields.summary;
                  const final = {id: re.id, key: re.key, name: re.fields.summary};
                  resultado.push(final)
               }
            }
            for (var resul of resultado) {
               await getWorklog(resul.key, id, true,worklogsUser,
usuario).then(responseW => {
                  UtilsService.addAllMap(worklogsUser, responseW);
               });
            }
            return worklogsUser;
         }, error => {
            throw Error('user or token invalid');
         });

      });
}

async function getIssuesWorkLogForUserV2(id, min, max, user) {
   const data = await CacheService.getCache(id);
   const usuario = user ? user : data.name;
   const jql = `(worklogAuthor in (${usuario})) AND (worklogDate >= "${min}" and
worklogDate <= "${max}")`;
   const fields =
`summary,worklog,issuetype,parent,project,status,assignee,reporter,aggregatetimes
pent,timeoriginalestimate,timeestimate`;
   return Promise.resolve('issuesWorklog')
      .then(result => {
         const config1 = {
            headers: {Authorization: `Basic ${tokenRobot}`}
         };
         let urlUserJira =
`${baseUrl}/search?jql=${jql}&fields=${fields}&maxResults=1000`;
         return axios.get(`${urlUserJira}`, config1).then(async resp => {
            const response = resp.data.issues;
```

```
            let resultado = response.filter(re=> re.fields.issuetype.id !== '10200');
            var worklogsUser = new Map();
            for (var issue of resultado) {
               let worklogs= await
getWorklogsPaginted(issue.fields.worklog,issue.key,id,min,max);
                worklogs = worklogs.filter(wl => wl.author.name === usuario);
               let responseW = await proccessDataWorklog(worklogsUser,worklogs,
true ,issue.key);
               UtilsService.addAllMap(worklogsUser, responseW);
            }
            return worklogsUser;
         }, error => {
            throw Error(error);
         });

      });
}

async function getIssuesWorkLogForUserNoIncurridos(id, min, max, users) {
   const data = await CacheService.getCache(id);
   let mapeosUsers = [];
   const jql = `(worklogAuthor in (${users.toString()})) AND (worklogDate >= "${min}"
and worklogDate <= "${max}")`;
   const fields =
`summary,worklog,issuetype,parent,project,status,assignee,reporter,aggregatetimes
pent,timeoriginalestimate,timeestimate`;
   return Promise.resolve('issuesWorklog')
      .then(result => {
         const config1 = {
            headers: {Authorization: `Basic ${tokenRobot}`}
         };
         let urlUserJira =
`${baseUrl}/search?jql=${jql}&fields=${fields}&maxResults=1000`;
         return axios.get(`${urlUserJira}`, config1).then(async resp => {
            const response = resp.data.issues;
            let resultado = response.filter(re=> re.fields.issuetype.id !== '10200');

            for(let usuario of users){
               let worklogsUser = new Map();
            for (let issue of resultado) {
               let worklogs= await
getWorklogsPaginted(issue.fields.worklog,issue.key,id,min,max);
                  worklogs = worklogs.filter(wl => wl.author.name === usuario);
                  let responseW = await proccessDataWorklog(worklogsUser,worklogs,
true ,issue.key);
                  UtilsService.addAllMap(worklogsUser, responseW);
               }
               mapeosUsers.push(worklogsUser);
            }
            return mapeosUsers;
         }, error => {
            throw Error('user or token invalid');
         });

      });
```

```
}

async function getEstimatedActual(id,min,max,projectKey,user) {
   const data = await CacheService.getCache(id);
   const name = user ? user :  data.name;
   const jql = `(worklogAuthor in (${name}) and worklogDate >= "${min}" and
worklogDate <= "${max}") and (project in (${projectKey}) )`;
   const fields = `worklog,project,parent,timeoriginalestimate`;
   const maxResults = 1000;

   return Promise.resolve('estimatedActual')
      .then(result => {
         const config1 = {
            headers: {Authorization: `Basic ${tokenRobot}`}
         };
         let                                  urlUserJira                                  =
`${baseUrl}/search?jql=${jql}&fields=${fields}&maxResults=${maxResults}`;
         return axios.get(`${urlUserJira}`, config1).then(async resp => {
            const response = resp.data.issues;
            const                evalueated                =                await
evaluateEstimatedActual(response,name,id,min,max);
           return  {
            estimated:  evalueated.estimated / 3600 + 'h',
            actual: evalueated.actual/3600 + 'h'
         };
         }, error => {
            throw Error('user or token invalid');
         });

      });
}

async function evaluateEstimatedActual(issues,usuario, id, fechaIni,fechaFin){
   let data = {estimated:0, actual:0};
   let worklogs = [];
   for (let issue of issues) {
      const      originalEstimate      =      issue.fields.timeoriginalestimate      ?
issue.fields.timeoriginalestimate : 0
      data.estimated += originalEstimate;
      worklogs      =      await      getWorklogsPaginted(issue.fields.worklog,issue.key,
id,fechaIni, fechaFin);
      for(let wk of worklogs.filter(wl => wl.author.name === usuario)){
         data.actual += wk.timeSpentSeconds
      }

   }
return data;
}

async function worklogsByissueAndUser(issues,usuario,id, fechaIni,fechaFin){
   let worklogs = [];
   for (let issue of issues) {
      worklogs.push(...await      getWorklogsPaginted(issue.fields.worklog,issue.key,
id,fechaIni, fechaFin));
   }
```

```
    return worklogs.filter(wl => wl.author.name === usuario);
}

async function getWorklogsPaginted(worklog,issueKey,id,fechaIni, fechaFin){
    let worklogs = worklog.worklogs;
    if(verificarPaginated(worklog)){
        let logs= [];
        let page = 0;
        let response = await getWorklogCustom(issueKey,id,page,fechaIni,fechaFin);
        logs = response.worklogs;
        if(verificarPaginated(response)){
            const pages =
UtilsService.returnStartAt(response.total,response.maxResults);

            do{
                page = page + 1;
                response = await getWorklogCustom(issueKey,id,page,fechaIni,fechaFin);
                logs.push(...response.worklogs);
            }while(page<= pages)
        }
        worklogs = logs;

    }
    return worklogs;
}

function verificarPaginated(worklog){
    return worklog.maxResults < worklog.total;
}

async function getWorklogCustom(issueKey,id,page,dateIni,dateFin) {
    const fechaIni =
UtilsService.momentMinusDaysMiliseconds(dateIni,"Europe/Madrid",1);
    const fechaFin =
UtilsService.momentAddDaysMiliseconds(dateFin,"Europe/Madrid",1);
    const promise = new Promise(async function (resolve, reject) {
        let responseWork  = await getWorklog2(issueKey,id,page,fechaIni,fechaFin);
        if (responseWork) {
            resolve(responseWork);
        }
        resolve(null);
    })
    return promise;

}

async function getAllWorklogs(issueKey,id,page,fechaIni,fechaFin,usuario){
  let responseWork  = await getWorklog2(issueKey,id,page,fechaIni,fechaFin);
  let worklogs= await
getWorklogsPaginted(responseWork,issueKey,id,fechaIni,fechaFin);
  worklogs = worklogs.filter(wl => wl.author.name === usuario);
  return worklogs;
}
```

```
module.exports = {
   addWorklog,
   addWorklog2,
   getUser,
   getIssues,
   getProyect,
   updateStatus,
   getStatusForIssue,
   getWorklog,
   getIssuesWorkLog,
   getIssuesWorkLogForUser,
   getEstimatedActual,
   getWorklogsPaginted,
   getIssuesWorkLogForUserV2,
   getIssuesWorkLogForUserNoIncurridos,
   getAllWorklogs,
   deleteWorklog
}
```

Figura 33
SERVICE MENU

```
"use strict";

const FestivoDao = require("../dao/festivos.dao");
const constantes = require("../constante");

function getMenus(rol) {
  return Promise.resolve("issues").then((result) => {
     const d2 = [
      [
        {
          text: `Tareas`,
          callback_data: JSON.stringify({
            command: "tareas",
          }),
          roles:["admin","manager","service","user"]
        },

        {
          text: `Mi info`,
          callback_data: JSON.stringify({
            command: "miInfo",
          }),
          roles:["admin","manager","service","user"]
        },
      ],
      [
        {
          text: `Incurrido masivo`,
          callback_data: JSON.stringify({
            command: "masivo",
          }),
          roles:["admin","manager","service","user"]
        },
```

```
        {
          text: `Incurrir festivos`,
          callback_data: JSON.stringify({
            command: "festivo",
          }),
          roles:["admin","manager","service","user"]
        }
      ],
      [
        {
          text: `Reportes`,
          callback_data: JSON.stringify({
            command: "reportes",
          }),
          roles:["admin","manager","service","user"]
        },
        {
          text: `Tutoriales`,
          callback_data: JSON.stringify({
            command: "tutoriales",
          }),
          roles:["admin","manager","service","user"]
        },
      ],
      [
          {
            text: `Equipo`,
            callback_data: JSON.stringify({
              command: "equipo",
            }),
            roles:["admin","manager","service"]
          }
        ]
    ];
    let filtrado = [];
    for(let men of d2){
      filtrado.push(men.filter(x=>x.roles.includes(rol)));
    }
    return filtrado;
  });
}

function getMeses() {
  return Promise.resolve("meses").then((result) => {
    var d2 = [];
    var meses = [
      "Enero",
      "Febrero",
      "Marzo",
      "Abril",
      "Mayo",
      "Junio",
      "Julio",
      "Agosto",
      "Setiembre",
```

```
      "Octubre",
      "Noviembre",
      "Diciembre",
    ];
    var grupo1 = [];
    var grupo2 = [];
    var grupo3 = [];
    for (var mes1 of meses.splice(0, 4)) {
      grupo1.push({
        text: mes1,
        callback_data: JSON.stringify({
          mes: mes1,
        }),
      });
    }
    for (var mes2 of meses.splice(0, 4)) {
      grupo2.push({
        text: mes2,
        callback_data: JSON.stringify({
          mes: mes2,
        }),
      });
    }
    for (var mes3 of meses.splice(0, 4)) {
      grupo3.push({
        text: mes3,
        callback_data: JSON.stringify({
          mes: mes3,
        }),
      });
    }
    d2.push(grupo1);
    d2.push(grupo2);
    d2.push(grupo3);
    return d2;
  });
}

function getMesesE() {
  return Promise.resolve("meses").then((result) => {
    var d2 = [];
    var meses = [
      "Enero",
      "Febrero",
      "Marzo",
      "Abril",
      "Mayo",
      "Junio",
      "Julio",
      "Agosto",
      "Setiembre",
      "Octubre",
      "Noviembre",
      "Diciembre",
    ];
```

```
    var grupo1 = [];
    var grupo2 = [];
    var grupo3 = [];
    for (var mes1 of meses.splice(0, 4)) {
      grupo1.push({
        text: mes1,
        callback_data: JSON.stringify({
          mesE: mes1,
        }),
      });
    }
    for (var mes2 of meses.splice(0, 4)) {
      grupo2.push({
        text: mes2,
        callback_data: JSON.stringify({
          mesE: mes2,
        }),
      });
    }
    for (var mes3 of meses.splice(0, 4)) {
      grupo3.push({
        text: mes3,
        callback_data: JSON.stringify({
          mesE: mes3,
        }),
      });
    }
    d2.push(grupo1);
    d2.push(grupo2);
    d2.push(grupo3);
    return d2;
  });
}
function obtenerConfig(msg) {
  const opts = {
    reply_to_message_id: msg.message_id,
    caption: `Hola <b>${msg.from.first_name} </b> soy un bot elige una de las opciones
disponibles`,
    reply_markup: {
      inline_keyboard: [
        [
          {
            text: "login in jira",
            callback_data: JSON.stringify({
              command: "login",
            }),
          },
        ],
      ],
    },
    parse_mode: "HTML",
  };
  return Promise.resolve(opts).then((result) => {
    return result;
  });
```

```
}

function listFestivos(hub) {
  return Promise.resolve("Festivos").then(async (result) => {
    let data = await FestivoDao.list();
    data = data.filter(fes=> fes.oficina.includes(hub))
    return data;
  });
}

module.exports = {
  getMenus,
  obtenerConfig,
  listFestivos,
  getMeses,
  getMesesE,
};
```

Figura 34
SERVICE TELEGRAM SERVICE

```
'use strict'
const axios = require('axios');
const JiraService = require('./jira.service');
const EmailService = require('./email.service');
const UtilsService = require("./utils.service");
const UserService = require("./users.service");
const EquipoService = require('./equipos.service');
const CacheService = require('./cache.service');

const KEY_LISTA = 'listEquipos';
function getPathDocument(idDocumento) {
   var urlObtenerPath =
`https://api.telegram.org/bot${process.env.KEY_TELE}/getFile?file_id=${idDocument
o}`;
    return axios.get(`${urlObtenerPath}`).then(resp => {
        const data = resp.data;
       if(!data.ok){
        throw Error('doccument id incorrect');
       }
        return data.result.file_path;
    }, error=>{
        throw Error('telegram token invalid');
    });




}

function getDocument(path){
    var url = `https://api.telegram.org/file/bot${process.env.KEY_TELE}/${path}`;
    return axios.get(url, {responseType: 'arraybuffer'})
    .then(res => {
        return res.data;
    })
}
```

```
async function reporteIncurrido(bot,id,fechIni,fechFin,usuarios, data, noLog){
   const response = {data: [], fechas:[]}
   const fechas = UtilsService.diasRangoFechasV2(fechIni, fechFin);
   const dataMasterExcel = [];
   for (var user of usuarios) {
      if(!noLog){
         bot.sendMessage(id, ` <b>Consultando usuario ${user?user:''}...</b>`, {
            parse_mode: 'HTML'
         });
      }

      let dataJira = [];
      let displayName = user;
      await JiraService.getIssuesWorkLogForUserV2(id, fechIni, fechFin,
user).then(async (result) => {
         for (var f of fechas) {
            var existe = result.has(f.fecha);
            if (existe) {
               var value = result.get(f.fecha);
               displayName = value.displayName;
               //dataJira.push({dia:f.fecha.substring(8,10),
value:`${UtilsService.vonvertirHoras(value.time)} (${value.sms})`});
               dataJira.push({
                  dia: f.fecha.substring(8, 10),
                  value: UtilsService.vonvertirHoras2(value.time),
                  sms:value.sms2
               });
            } else {
               dataJira.push({dia: f.fecha.substring(8, 10), value: 0, sms:''});
            }
         }
         // dataJira.sort((a, b) => a.dia - b.dia);
         const dataExcel = [displayName];
         for (var x of dataJira) {
            dataExcel.push(x.value+'|'+x.sms)
         }
         dataMasterExcel.push(dataExcel);

      });

   }
   response.data = dataMasterExcel;
   response.fechas = fechas;
   return response;
}

async function reporteNoIncurrido(id){
   const response = {data: [], fechas:[]}
   const fecha = UtilsService.convertirMomentString(UtilsService.fechaActual());
   const listaEquipos = await EquipoService.listEquipos();
   const dataMasterExcel = [];
   let dataJira = [];
   for (const equipo of listaEquipos) {
      if (equipo.alertTeams) {
```

```
        console.log("Equipo: ", equipo);
        for (const user of equipo.peoples) {
            await JiraService.getIssuesWorkLogForUserV2(id, fecha, fecha,
user).then(async (result) => {
                    var existe = result.has(fecha);
                    if (existe) {
                        var value = result.get(fecha);
                        if(value.user !== user){

dataJira.push({usuario:user,correo:user+'@emeal.nttdata.com',fecha,hora:value.time
,mensaje:'Hasta el momento no ha incurrido'});
                        }else if(value.user === user && value.time < 32400){

dataJira.push({usuario:user,correo:user+'@emeal.nttdata.com',fecha,hora:value.time
,mensaje:`le faltan incurrir ${(32400 - value.time)/3600}h`});
                        }else if(value.user === user && value.time > 32400){

dataJira.push({usuario:user,correo:user+'@emeal.nttdata.com',fecha,hora:value.time
,mensaje:`tienes  ${(value.time - 32400)/3600}h de más incurridas, si esto se trata de
un error revisar su incurrido`});
                        }

                    } else {

dataJira.push({usuario:user,correo:user+'@emeal.nttdata.com',fecha,hora:0,mensaje
:'Hasta el momento no ha incurrido'});
                    }
            });
        }
    }

    }
    const dataExcel = [];
    for (var x of dataJira) {
        dataExcel.push([x.usuario,x.correo,x.fecha,(x.hora/3600),x.mensaje])
    }
    dataMasterExcel.push(dataExcel);
    response.data = dataMasterExcel;
    return response;
}

async function reporteNoIncurridoV2(id, peoples){
    const response = {data: [], fechas:[]}
    const fecha = UtilsService.convertirMomentString(UtilsService.fechaActual());
    const dataMasterExcel = [];
    let dataJira = [];
        for (const user of peoples) {
            await JiraService.getIssuesWorkLogForUserV2(id, fecha, fecha,
user).then(async (result) => {
                    var existe = result.has(fecha);
                    if (existe) {
                        var value = result.get(fecha);
                        if(value.user !== user){
```

```
dataJira.push({usuario:user,correo:user+'@emeal.nttdata.com',fecha,hora:value.time
,mensaje:'Hasta el momento no ha incurrido'});
                    }else if(value.user === user && value.time < 32400){

dataJira.push({usuario:user,correo:user+'@emeal.nttdata.com',fecha,hora:value.time
,mensaje:`le faltan incurrir ${(32400 - value.time)/3600}h`});
                    }else if(value.user === user && value.time > 32400){

dataJira.push({usuario:user,correo:user+'@emeal.nttdata.com',fecha,hora:value.time
,mensaje:`tienes  ${(value.time - 32400)/3600}h de más incurridas, si esto se trata de
un error revisar su incurrido`});
                    }

                } else {

dataJira.push({usuario:user,correo:user+'@emeal.nttdata.com',fecha,hora:0,mensaje
:'Hasta el momento no ha incurrido'});
                }
        });
    }

  const dataExcel = [];
  for (var x of dataJira) {
    dataExcel.push([x.usuario,x.correo,x.fecha,(x.hora/3600),x.mensaje])
  }
  dataMasterExcel.push(dataExcel);
  response.data = dataMasterExcel;
  return response;
}

async function messageUpdateUsers(bot,id){
  UserService.listAll().then(response=>{
    console.log(response);
    let message = `Hola <b> %%NOMBRE%%</b> hubo\n`;
    message += `<b>ACTUALIZACIONES EN EL BOT</b> \n`;
    message += `<b>1. </b> Eliminar worklogs de forma masiva \n`
    message += `<b>Instrucciones </b> \n`
    message += `<b>1. </b> Debe de seleccionar el boton <b>'Eliminar
worklogs'</b> \n`
    message += `<b>2. </b> Seleccione la opción <b>'Descargar formato'</b>, debe
de ingresar el código de tu tarea o si son mas de una separar por comas ejm 'ABC-
123,BBB-123' \n`
    message += `<b>3. </b> Se descargara un excel con tus worklogs de tus tareas,
seleccione los worklogs que desea eliminar poniendo la opción SI en la columna
eliminar \n`
    message += `<b>4. </b> Seleccione la opción <b>'Subir formato'</b>, adjunte el
excel modificado. Listo el bot procesará la información \n`
    message += `<b>SALUDOS</b>`
for(const user of response){
  let mensajeNuevo = message.replace("%%NOMBRE%%",user.displayName);
  bot.sendMessage(user.idTelegram, mensajeNuevo, {
    parse_mode: 'HTML'
  });
}
```

```
bot.sendMessage(id, "Mensaje enviado a todos los usuarios...", {
   parse_mode: 'HTML'
});

   })
}

async function autoIncurridosEquipos(id){
   const fecha = UtilsService.convertirMomentString(UtilsService.fechaActual());
   const listaUsuarios = await UserService.listAll();
   await CacheService.setCache(KEY_LISTA,JSON.stringify([]));
   let dataJira = [];
  // const xxx = listaUsuarios.splice(0,5);
   for (const user of listaUsuarios ) {
      if (!user.notAutomatedIncurred) {
       console.log("User: ", user.name);
          await JiraService.getIssuesWorkLogForUserV2(user.idTelegram, fecha,
fecha, user.name).then(async (result) => {
                    var existe = result.has(fecha);
                    if (existe) {
                       var value = result.get(fecha);
                       if(value.user !== user.name){

dataJira.push({user,fecha,incurrir:'9h',id:user.idTelegram,name:user.displayName});
                       }else if(value.user === user.name && value.time < 32400){
                          dataJira.push({user,fecha,incurrir:`${(32400 -
value.time)/3600}h`,id:user.idTelegram,name:user.displayName});
                       }

                    } else {

dataJira.push({user,fecha,incurrir:'9h',id:user.idTelegram,name:user.displayName});
                    }
          });
       }

   }

   for (var x of dataJira) {
      const issueDefault = await obtenerDefaultIssue(x.user.name)
      if(issueDefault){
        await JiraService.addWorklog(x.incurrir, x.id, issueDefault,
"AUTOINCURRIDO", fecha).then(async (res) => {
            await EmailService.sendEmail(x.user.name+'@emeal.nttdata.com',
'AUTOINCURRIDO ' + fecha, `<span style="font-weight: bold;">Hola, ${x.name}
</span>
            <p>Se ha incurrido de forma automatica en la siguiente tarea <a
href="https://umane.emeal.nttdata.com/jiraito/browse/${issueDefault}">${issueDefaul
t}</a><br> Saludos</p>`)
         }, err => {

         });

      }
```

```
    }

}

async function obtenerDefaultIssue(usuario){
let data = await CacheService.getCache2(KEY_LISTA);
let defaultIssue = null;
if(data != null ){
   let equipos = JSON.parse(data);
   const user = equipos.find(x=>x.user == usuario);
   if(user){
      defaultIssue = user.issue;
   }else{
      const listaEquipos = await EquipoService.findEquipoByUser(usuario);
      if(listaEquipos.length > 0){
      const peoples = listaEquipos[0].peoples.map(x=>{return
{user:x,issue:listaEquipos[0].issueDefault}});
      equipos.push(...peoples);
      await CacheService.setCache(KEY_LISTA,JSON.stringify(equipos));
      defaultIssue = listaEquipos[0].issueDefault;
      }
   }
}
return defaultIssue;
}

async function dataExcelEliminarWorklog(worklogs){
   let dataExcel = [];
   for (let x of worklogs) {
     var fecha =
UtilsService.converterFechaTimezone4(x.started,x.updateAuthor.timeZone);

dataExcel.push([x.id,x.issue,fecha,(x.timeSpentSeconds/3600)+'h',x.comment,'NO'])
   }
   return dataExcel;
}

async function reporteExcelGrupoTeams(){
   const response = {data: [], fechas:[]}
   const listaEquipos = await EquipoService.listEquipos();
   const dataMasterExcel = [];
   let data = [];
   for (const equipo of listaEquipos) {
     if (equipo.idGroupTeams) {
      console.log("Equipo: ", equipo);

data.push({equipo:`${equipo.key}_${equipo.usernameServiceLeader}`,idGrupo:equip
o.idGroupTeams,path:equipo.pathOneDrive});
   }

   }
   const dataExcel = [];
   for (var x of data) {
      dataExcel.push([x.equipo,x.idGrupo,x.path])
   }
```

```
    dataMasterExcel.push(dataExcel);
    response.data = dataMasterExcel;
    return response;
}

module.exports = {
    getPathDocument,
    getDocument,
    reporteIncurrido,
    messageUpdateUsers,
    reporteNoIncurrido,
    autoIncurridosEquipos,
    dataExcelEliminarWorklog,
    reporteExcelGrupoTeams,
    reporteNoIncurridoV2
}
```

Figura 35
SERVICE USERS

```
'use strict'
const UsersDao = require('../dao/users.dao');



async function updateUsers(filter, setter){
   const result = await UsersDao.updateUsers(filter, setter);
    return result;
}

async function getUser(key){
    return   await UsersDao.getUsers(key);
}

async function listAll(key){
    return   await UsersDao.listAllUsers();
}

async function verificarToken(name, token){
    const user = await UsersDao.getUserByName(name);
    if(user.token === token){
        await UsersDao.updateUsers({name:name}, {active:true});
        return true;
    }
    return   false;;
}



module.exports = {
    updateUsers,
    getUser,
    listAll,
    verificarToken
}
```

Figura 36
SERVICE UTILS

```
"use strict";
const constantes = require("../constante");
const moment = require("moment-timezone");
const axios = require("axios");
var jsJoda = require("@js-joda/core");
require("@js-joda/timezone");

function converte(text) {
  const s = 1;
  const m = 60;
  const h = 3600;
  const d = 86400;
  if (text.includes("s")) {
    const segundo = text.split("s");
    return segundo[0] * s;
  } else if (text.includes("m")) {
    const minuto = text.split("m");
    return minuto[0] * m;
  } else if (text.includes("h")) {
    const hora = text.split("h");
    return hora[0] * h;
  } else if (text.includes("d")) {
    const dia = text.split("d");
    return dia[0] * d;
  } else {
    return 8 * h;
  }
}

function sumarDias(fecha, dias) {
  fecha.setDate(fecha.getDate() + dias);
  return fecha;
}

function traslateLine(line) {
  const lineArray = line.split(";");
  if (lineArray.length > 1 && lineArray[0] !== "") {
    const data = {
      fecha: darFormatoFecha(lineArray[0].trim()),
      hora: lineArray[1],
      jirakey: lineArray[2],
      comentarios: lineArray[3],
    };
    return data;
  }
  if (lineArray[0] === "") {
    return null;
  }
}

function skipFirstLine(lines) {
  return lines.splice(1, lines.length);
}
```

```
function darFormatoFecha(fecha) {
  const fechas = fecha.split("/");
  if (fechas.length === 3) {
    return fechas[2] + "-" + fechas[1] + "-" + fechas[0];
  }
  return null;
}

function formatDateString1(fecha) {
  return (
    fecha.getFullYear() +
    "-" +
    zfill(fecha.getMonth() + 1, 2) +
    "-" +
    zfill(fecha.getDate(), 2)
  );
}

function vonvertirHoras(hora) {
  const h = 3600;
  if (hora === undefined || hora === 0) {
    return "0h";
  }
  return hora / h + "h";
}

function vonvertirHoras2(hora) {
  const h = 3600;
  if (hora === undefined || hora === 0) {
    return 0;
  }
  return hora / h;
}

function diasDeMeses(anio, mesI) {
  var listFechas = [];
  var meses = [
    "",
    "Enero",
    "Febrero",
    "Marzo",
    "Abril",
    "Mayo",
    "Junio",
    "Julio",
    "Agosto",
    "Setiembre",
    "Octubre",
    "Noviembre",
    "Diciembre",
  ];
  anio = anio === undefined ? new Date().getFullYear() : anio;
  var mes = meses.indexOf(mesI);
```

```
  var diasMes = new Date(anio, mes, 0).getDate();
  var diasSemana = ["D", "L", "M", "M", "J", "V", "S"];

  for (var dia = 1; dia <= diasMes; dia++) {
    var data = { fecha: "", day: "" };
    var indice = new Date(anio, mes - 1, dia).getDay();
    data.fecha = anio + "-" + zfill(mes, 2) + "-" + zfill(dia, 2);
    data.day = diasSemana[indice];
    listFechas.push(data);
  }
  return listFechas;
}

function zfill(number, width) {
  var numberOutput = Math.abs(number); /* Valor absoluto del número */
  var length = number.toString().length; /* Largo del número */
  var zero = "0"; /* String de cero */

  if (width <= length) {
    if (number < 0) {
      return "-" + numberOutput.toString();
    } else {
      return numberOutput.toString();
    }
  } else {
    if (number < 0) {
      return "-" + zero.repeat(width - length) + numberOutput.toString();
    } else {
      return zero.repeat(width - length) + numberOutput.toString();
    }
  }
}

function addAllMap(target, source) {
  if (target instanceof Map) {
    Array.from(source.entries()).forEach((it) => target.set(it[0], it[1]));
  } else if (target instanceof Set) {
    source.forEach((it) => target.add(it));
  }
}

function getDocumentByURL(url) {
  return axios.get(url, { responseType: "arraybuffer" }).then((res) => {
    return res.data;
  });
}
function converterFechaTimezone(fecha, timezone) {
  let date = new Date(fecha);
  let formatter = new Intl.DateTimeFormat("en-GB", { timeZone: timezone });
  let usDate = formatter.format(date);
  const anio = usDate.substring(6, 10);
  const mes = usDate.substring(3, 5);
  const dia = usDate.substring(0, 2);
  return anio + "-" + mes + "-" + dia;
}
```

```
function converterFechaTimezone2(startDate, timezone) {
  let { LocalDateTime, ZoneId, ZonedDateTime } = jsJoda;
  const startDateC = startDate.substring(0, 19);
  const dateAtZone = LocalDateTime.parse(startDateC)
    .atZone(ZoneId.of(timezone))
    .toString();

  /**const dateAtInstant = ZonedDateTime
    .parse(dateAtZone)
    .withZoneSameInstant(ZoneId.of(timezone))
    .toString() **/

  const localDate = ZonedDateTime.parse(dateAtZone).withZoneSameLocal(
    ZoneId.of(timezone)
  );
  return (
    localDate.year() +
    "-" +
    zfill(localDate.monthValue(), 2) +
    "-" +
    zfill(localDate.dayOfMonth(), 2)
  );
}

function converterFechaTimezone3(startDate, timezone) {
  let fecha = new Date(startDate).toLocaleString("en-GB", {
    timeZone: timezone,
  });
  return (
    fecha.substring(6, 10) +
    "-" +
    fecha.substring(3, 5) +
    "-" +
    fecha.substring(0, 2)
  );
}

function converterFechaTimezone4(startDate, timezone) {
  let fecha = new Date(startDate).toLocaleString("en-GB", {
    timeZone: timezone,
  });
  return fecha;
}

/**
 * @param ini tiene que ir en formato yyyy-mm-dd
 * @param fin tiene que ir en formato yyyy-mm-dd
 * */
function diasRangoFechas(ini, fin) {
  const timezone = "America/Lima";
  var fechaInicio = moment(ini).tz(timezone);
  var fechIni = fechaInicio.toDate().valueOf();
  var fechaFin = moment(fin).tz(timezone).toDate().valueOf();
  var diasSemana = ["D", "L", "M", "M", "J", "V", "S"];
```

```
  var listFechas = [];
  var diff = fechaFin - fechIni;
  var dias = diff / (1000 * 60 * 60 * 24);
  for (var dia = 1; dia <= dias + 1; dia++) {
    var data = { fecha: "", day: "" };
    if (dia !== 1) {
      fechaInicio = fechaInicio.add(1, "d");
    }
    var indice = fechaInicio.day();
    data.fecha =
      fechaInicio.year() +
      "-" +
      zfill(fechaInicio.month() + 1, 2) +
      "-" +
      zfill(fechaInicio.date(), 2);
    data.day = diasSemana[indice];
    listFechas.push(data);
  }
  return listFechas;
}

function diasRangoFechasV2(ini, fin) {
  const timezone = "Europe/Madrid";
  let { LocalDateTime, ZoneId, ZonedDateTime, ChronoUnit } = jsJoda;
  const d1 = LocalDateTime.parse(ini + "T00:00")
    .atZone(ZoneId.of(timezone))
    .toString();
  const l1 = ZonedDateTime.parse(d1).withZoneSameLocal(ZoneId.of(timezone));

  const d2 = LocalDateTime.parse(fin + "T00:00")
    .atZone(ZoneId.of(timezone))
    .toString();
  const l2 = ZonedDateTime.parse(d2).withZoneSameLocal(ZoneId.of(timezone));
  let fechaInicio = l1;
  var listFechas = [];
  const dias = l1.until(l2, ChronoUnit.DAYS);
  for (var dia = 0; dia <= dias; dia++) {
    var data = { fecha: "", day: "" };
    if (dia !== 0) {
      fechaInicio = fechaInicio.plusDays(1);
    }
    data.fecha =
      fechaInicio.year() +
      "-" +
      zfill(fechaInicio.monthValue(), 2) +
      "-" +
      zfill(fechaInicio.dayOfMonth(), 2);
    data.day = inicialDia(fechaInicio.dayOfWeek().name());
    listFechas.push(data);
  }
  return listFechas;
}

function diasRangoFechasCsv(ini, fin) {
  const timezone = "Europe/Madrid";
```

```
  let { LocalDateTime, ZoneId, ZonedDateTime, ChronoUnit } = jsJoda;
  const d1 = LocalDateTime.parse(ini + "T00:00")
    .atZone(ZoneId.of(timezone))
    .toString();
  const l1 = ZonedDateTime.parse(d1).withZoneSameLocal(ZoneId.of(timezone));

  const d2 = LocalDateTime.parse(fin + "T00:00")
    .atZone(ZoneId.of(timezone))
    .toString();
  const l2 = ZonedDateTime.parse(d2).withZoneSameLocal(ZoneId.of(timezone));
  let fechaInicio = l1;
  var listFechas = [];
  const dias = l1.until(l2, ChronoUnit.DAYS);
  for (var dia = 0; dia <= dias; dia++) {
    var data = { fecha: "", day: "" };
    if (dia !== 0) {
      fechaInicio = fechaInicio.plusDays(1);
    }
    data.fecha =
      zfill(fechaInicio.dayOfMonth(), 2) +
      "/" +
      zfill(fechaInicio.monthValue(), 2) +
      "/" +
      fechaInicio.year();
    data.day = inicialDia(fechaInicio.dayOfWeek().name());
    listFechas.push(data);
  }
  return listFechas;
}

function returnStartAt(total, maxResults) {
  let pages = Math.trunc(total / maxResults);
  const residuo = total % maxResults;
  return residuo !== 0 ? (pages += 1) : pages;
}
function converterDateMiliseconds(date, timezone) {
  let mili = moment(date).tz(timezone);
  return mili.valueOf();
}

function momentAddDaysMiliseconds(date, timezone, days) {
  let mili = moment(date).tz(timezone);
  return mili.add(days, "d").valueOf();
}

function momentMinusDaysMiliseconds(date, timezone, days) {
  let mili = moment(date).tz(timezone);
  return mili.subtract(days, "d").valueOf();
}

function mesActual() {
  var meses = [
    "Enero",
    "Febrero",
    "Marzo",
```

```
    "Abril",
    "Mayo",
    "Junio",
    "Julio",
    "Agosto",
    "Setiembre",
    "Octubre",
    "Noviembre",
    "Diciembre",
  ];
  let day = moment().tz("America/Lima");
  return meses[day.toDate().getMonth()];
}

function fechaActual() {
  let day = moment().tz("America/Lima");
  return day;
}

function convertirMomentString(fecha) {
  return (
    fecha.year() +
    "-" +
    zfill(fecha.month() + 1, 2) +
    "-" +
    zfill(fecha.date(), 2)
  );
}

function convertFechaReporte(fecha) {
  const timezone = "Europe/Madrid";
  moment.locale("es");
  let fechaInicio = moment(fecha).tz(timezone);
  const listFech = fechaInicio.format("LL").split("de");
  return (
    zfill(listFech[0], 2) +
    "-" +
    listFech[1].replace(" ", "").substring(0, 3).toUpperCase()
  );
}

function eliminarNumeros(text) {
  return text.replace(/\d+/g, "");
}

function eliminarUltimaLetra(text) {
  return text ? text.substr(0, text.length - 1) : text;
}

function inicialDia(day) {
  let myMap = new Map();
  myMap.set("SUNDAY", "D");
  myMap.set("MONDAY", "L");
  myMap.set("TUESDAY", "M");
  myMap.set("WEDNESDAY", "M");
```

```
  myMap.set("THURSDAY", "J");
  myMap.set("FRIDAY", "V");
  myMap.set("SATURDAY", "S");

  return myMap.get(day);
}

function generateClave() {
  var pass = '';
  var str = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ' +
        'abcdefghijklmnopqrstuvwxyz0123456789';

  for (let i = 1; i <= 8; i++) {
     var char = Math.floor(Math.random()
               * str.length + 1);

     pass += str.charAt(char)
  }

  return pass;
}

function verifircarPermisoAdmin(rol){
  const roles = ["admin","manager","service"];
  return roles.includes(rol);
}

module.exports = {
  converte,
  sumarDias,
  traslateLine,
  skipFirstLine,
  darFormatoFecha,
  vonvertirHoras,
  diasDeMeses,
  formatDateString1,
  addAllMap,
  getDocumentByURL,
  diasRangoFechas,
  converterFechaTimezone,
  converterFechaTimezone2,
  returnStartAt,
  converterDateMiliseconds,
  momentAddDaysMiliseconds,
  momentMinusDaysMiliseconds,
  mesActual,
  fechaActual,
  convertirMomentString,
  vonvertirHoras2,
  convertFechaReporte,
  eliminarNumeros,
  eliminarUltimaLetra,
  inicialDia,
  diasRangoFechasV2,
  converterFechaTimezone3,
```

```
  diasRangoFechasCsv,
  converterFechaTimezone4,
  generateClave,
  verifircarPermisoAdmin
};
```

Figura 37
CONTROLLER INDEX

```
"use strict";
require("dotenv").config();

const TelegramBot = require("node-telegram-bot-api");
const MenuService = require("./services/menu.service");
const UtilsService = require("./services/utils.service");
const JiraService = require("./services/jira.service");
const CacheService = require("./services/cache.service");
const TelegramService = require("./services/telegram.service");
const UsersService = require("./services/users.service");
const ExcelService = require("./services/excel.service");
const EquipoService = require("./services/equipos.service");
const EmailService = require("./services/email.service");
const PeopleService = require("./services/people.service");
const YoutubeService = require("./services/youtube.service");
const constantes = require("./constante");
const schedule = require("node-schedule");
//const { svg2png } = require("svg-png-converter");
const { TEMPLATE_EMAIL } = require("./template");

let regexpKeyboard = /(hola|Hola|HOLA|ola|OLA|Hello|hello|HELLO)/g;

const bot = new TelegramBot(process.env.KEY_TELE, {
  polling: true,
});

const urlFormato = constantes.URL_FORMATO_CSV;

const logoBase64 = constantes.LOGO;

bot.onText(/\/start/, (msg, [source, match]) => {
  const chatId = msg.chat.id;
  let buff = Buffer.from(logoBase64, "base64");
  MenuService.obtenerConfig(msg)
    .then((result) => {
      bot.sendPhoto(msg.chat.id, buff, result);
    })
    .catch((error) => bot.sendMessage(chatId, "Not found"));
});

bot.onText(regexpKeyboard, async (msg, match) => {
  const chatId = msg.chat.id;
  const data = await CacheService.getCache(chatId);
  if (data && data.active) {
  await  MenuService.getMenus(data.rol).then((result) => {
      bot.sendMessage(
```

```
        chatId,
        `Hola <b>${data.displayName} </b> elige una de las opciones`,
        {
          reply_markup: {
            inline_keyboard: result,
          },
          parse_mode: "HTML",
        }
      );
    });
  } else {
    let buff = Buffer.from(logoBase64, "base64");
    MenuService.obtenerConfig(msg)
      .then((result) => {
        bot.sendPhoto(msg.chat.id, buff, result);
      })
      .catch((error) => bot.sendMessage(chatId, "Not found"));
  }
});

bot.on("callback_query", async function onCallbackQuery(callbackQuery) {
  const data = JSON.parse(callbackQuery.data);
  const msg = callbackQuery.message;
  const {
    chat: { id },
  } = msg;
  const opts = {
    chat_id: msg.chat.id,
    message_id: msg.message_id,
  };
  if (data.command === "login") {
    bot
      .sendMessage(opts.chat_id, "Ingrese <b>Usuario Corto nttdata</b>", {
        reply_markup: {
          force_reply: true,
        },
        parse_mode: "HTML",
      })
      .then(async (response) => {
        const replyListenerId = bot.onReplyToMessage(
          response.chat.id,
          response.message_id,
          async (msg) => {
            bot.removeReplyListener(replyListenerId);
            const {
              chat: { id },
            } = msg;
            const correo = `${msg.text}@emeal.nttdata.com`;
            bot.sendMessage(id, ` <b>Se ha enviado un correo electronico a
<code>${correo}</code> con un código de verificación</b>`, {
              parse_mode: "HTML",
            });
            JiraService.getUser(msg.text, id)
              .then(async (data) => {
```

```
            await EmailService.sendEmailRobot(correo, 'Código verificacion bot
Telegram', TEMPLATE_EMAIL({nombres:data.displayName, token:data.token}));
            bot.sendMessage(opts.chat_id, "Ingrese <b>Código de verificación</b>", {
               reply_markup: {
                 force_reply: true,
               },
            parse_mode: "HTML",
    }).then(respCodigo=>{
      const replyListenerId = bot.onReplyToMessage(
        respCodigo.chat.id,
        respCodigo.message_id,
        async (msg) => {
          bot.removeReplyListener(replyListenerId);
          const {
            chat: { id },
          } = msg;
          bot.sendMessage(id, ` <b>Validando...</b>`, {
            parse_mode: "HTML",
          });
         const valido = await UsersService.verificarToken(data.name, msg.text);
         if(valido){
          data.active = true;
          CacheService.setCache(`${id}`, data, 3600 * 1000);
          MenuService.getMenus(data.rol).then((result) => {
            bot.sendMessage(
              id,
              `Hola <b>${data.displayName} </b> elige una de las opciones`,
              {
                reply_markup: {
                  inline_keyboard: result,
                },
                parse_mode: "HTML",
              }
            );
          });
         }else{
          bot.sendMessage(opts.chat_id, `<b>Código Invalido vuelva a intentarlo</b>`,
{
            parse_mode: "HTML",
          })
         }
         bot.answerCallbackQuery(callbackQuery.id);
        }
      );
    })})
            .catch((error) =>
              bot.sendMessage(opts.chat_id, `<b>${error}</b>`, {
                parse_mode: "HTML",
              })
            );
            bot.answerCallbackQuery(callbackQuery.id);
        }
      );
    });
  } else if (data.command === "tareas") {
```

```
  const {
    chat: { id },
  } = msg;
  bot.sendMessage(id, ` <b>Consultando...</b>`, {
    parse_mode: "HTML",
  });
  JiraService.getIssues(msg, id).then((result) => {
    if (result.length > 0) {
      for (var res of result) {
        bot.sendMessage(id, res.mensaje, {
          reply_markup: {
            inline_keyboard: res.menu,
          },
          parse_mode: "HTML",
        });
      }
    } else {
      bot.sendMessage(id, ` <b>No cuenta con ninguna tarea asignada</b>`, {
        parse_mode: "HTML",
      });
    }
  });
  bot.answerCallbackQuery(callbackQuery.id);
} else if (data.command === "proyectos") {
  const {
    chat: { id },
  } = msg;
  bot.sendMessage(id, ` <b>Consultando...</b>`, {
    parse_mode: "HTML",
  });
  const data = await CacheService.getCache(id);

  JiraService.getProyect(msg, id).then((result) => {
    bot.sendMessage(
      id,
      `Hola <b>${data.displayName}  </b> estos son tus proyectos en jira`,
      {
        parse_mode: "HTML",
      }
    );
    for (var res of result) {
      bot.sendMessage(id, `${res.key} - ${res.name}`, {
        reply_markup: {
          inline_keyboard: [
            [
              {
                text: `Info`,
                callback_data: JSON.stringify({
                  key: "infoP",
                  keyp: res.key,
                }),
              },
            ],
          ],
        },
```

```
        parse_mode: "HTML",
      });
    }
  });
  bot.answerCallbackQuery(callbackQuery.id);
} else if (data.command === "masivo") {
  const opts = {
    reply_markup: {
      inline_keyboard: [
        [{ text: "Descargar formato", url: urlFormato }],
        [
          {
            text: `Subir Formato`,
            callback_data: JSON.stringify({
              command: "subir",
            }),
          },
        ],
      ],
    },
    parse_mode: "HTML",
  };

  bot.sendMessage(id, "<b>Elige una de las opciones</b>", opts);
} else if (data.command === "subir") {
  bot.sendMessage(opts.chat_id, "sube el formato.csv", {
     reply_markup: {
      force_reply: true,
    },
   }).then(async (response) => {
     const replyListenerId = bot.onReplyToMessage(
      response.chat.id,
      response.message_id,
      async (msg) => {
        bot.removeReplyListener(replyListenerId);
        const {
          chat: { id },
        } = msg;
        if (
          msg.document === undefined ||
          msg.document.mime_type !== "text/csv"
        ) {
          bot.sendMessage(
            id,
            ` <b>formato incorrecto vuelva a intentar</b>`,
            {
              parse_mode: "HTML",
            }
          );
        } else {
          bot.sendMessage(id, ` <b>Procesando...</b>`, {
            parse_mode: "HTML",
          });
        }
        TelegramService.getPathDocument(msg.document.file_id).then(
```

```
            (response) => {
              TelegramService.getDocument(response).then((res) => {
                const newBuffer = Buffer.from(res);
                const bufferString = newBuffer.toString("latin1");
                const lines = UtilsService.skipFirstLine(
                  bufferString.split(/\r?\n/)
                );
                const linesConvertido = lines
                  .map(UtilsService.traslateLine)
                  .filter((x) => x !== null);

                for (var tarea of linesConvertido) {
                  const fechIncurrir = tarea.fecha;
                  JiraService.addWorklog2(
                    tarea.hora,
                    id,
                    tarea.jirakey,
                    tarea.comentarios,
                    tarea.fecha
                  ).then(
                    (res) => {
                      bot.sendMessage(
                        id,
                        ` <b>Fecha: ${fechIncurrir}</b> exitoso`,
                        {
                          parse_mode: "HTML",
                        }
                      );
                    },
                    (err) => {
                      bot.sendMessage(
                        id,
                        ` <b>Fecha: ${fechIncurrir}</b> error`,
                        {
                          parse_mode: "HTML",
                        }
                      );
                    }
                  );
                }
                bot.sendMessage(id, ` <b>Carga Masiva Finalizada</b>`, {
                  parse_mode: "HTML",
                });
                bot.answerCallbackQuery(callbackQuery.id);
              });
            },
            (error) => {}
          );
        }
      );
    });
  } else if (data.command === "festivo") {
    const data = await UsersService.getUser(id);
    if (!data.isFestivo) {
      bot
```

```
      .sendMessage(opts.chat_id, "Ingrese id del jira", {
        reply_markup: {
          force_reply: true,
        },
      })
      .then(async (response) => {
        const replyListenerId = bot.onReplyToMessage(
          response.chat.id,
          response.message_id,
          async (msg) => {
            bot.removeReplyListener(replyListenerId);
            const {
              chat: { id },
            } = msg;
            if (msg.text !== "" && msg.text !== undefined) {
              let people = {hub:'TRU'};
              if(!data.hub){
                people = await
PeopleService.obtenerPersonaByUsuario(data.name.toUpperCase());
              }
              MenuService.listFestivos(people.hub).then((respFes) => {
                let existeError = true;
                for (var tarea of respFes) {
                  const fechIncurrir = tarea.fecha;
                  const comentario = tarea.comentarios;
                  JiraService.addWorklog2(
                    tarea.hora,
                    id,
                    msg.text,
                    comentario,
                    tarea.fecha
                  ).then(
                    (res) => {
                      bot.sendMessage(
                        id,
                        `<b>Fecha: ${fechIncurrir}</b> <b>Comentario: ${comentario}</b>
exitoso`,
                        {
                          parse_mode: "HTML",
                        }
                      );
                      existeError = false;
                    },
                    (err) => {
                      console.log(err);
                      existeError = true;
                      bot.sendMessage(
                        id,
                        `<b>Fecha: ${fechIncurrir}</b> error`,
                        {
                          parse_mode: "HTML",
                        }
                      );
                    }
                  );
```

```
              }
              if (!existeError) {
                const updateUser = UsersService.updateUsers(
                  { idTelegram: id },
                  { isFestivo: true }
                );
              }
            });
          } else {
            bot.sendMessage(id, ` <b>Ingrese jira key</b>`, {
              parse_mode: "HTML",
            });
          }
          bot.answerCallbackQuery(callbackQuery.id);
        }
      );
    });
  } else {
    bot.sendMessage(id, ` <b>UD. Ya incurrio los festivos</b>`, {
      parse_mode: "HTML",
    });
    bot.answerCallbackQuery(callbackQuery.id);
  }
} else if (data.command === "cambiarE") {
  JiraService.getStatusForIssue(data.issueKey, id).then((result) => {
    bot.sendMessage(
      opts.chat_id,
      `Seleccione un estado para <b> ${data.issueKey}</b>` + "✅",
      {
        reply_markup: {
          keyboard: result,
          resize_keyboard: true,
          one_time_keyboard: true,
        },
        parse_mode: "HTML",
      }
    );
  });
} else if (data.command === "wlIssue") {
  JiraService.getWorklog(data.issueKey, id, false).then((result) => {
    var mensaje = `<b>Incurrido de la tarea ${data.issueKey}</b> \n`;
    for (const [key, value] of result) {
      mensaje += `<code>${key} = ${UtilsService.vonvertirHoras(
        value.time
      )}</code> (${value.sms})  \n`;
    }
    bot.sendMessage(opts.chat_id, mensaje, {
      parse_mode: "HTML",
    });
  });
} else if (data.command === "reportes") {
  const opts = {
    reply_markup: {
      inline_keyboard: [
        [
```

```
          {
            text: `Excel`,
            callback_data: JSON.stringify({
              reporte: "excel",
            }),
          },
          {
            text: `SMS`,
            callback_data: JSON.stringify({
              reporte: "sms",
            }),
          },
        ],
      ],
    },
    parse_mode: "HTML",
  };
  bot.sendMessage(id, "<b>Elige una de las opciones</b>", opts);
} else if (data.command === "miInfo") {
  bot.sendMessage(id, `<b>Consultando información</b>`, {
    parse_mode: "HTML",
  });
  infoPersona(bot, id);
  bot.answerCallbackQuery(callbackQuery.id);
} else if (data.key === "infoP") {
  let fechas = UtilsService.diasDeMeses(undefined, UtilsService.mesActual());
  let primero = fechas[0].fecha;
  let ultimo = fechas[fechas.length - 1].fecha;
  let mensaje = "";
  JiraService.getEstimatedActual(id, primero, ultimo, data.keyp).then(
    (result) => {
      mensaje += `<b> Proyecto ${data.keyp} </b>  \n`;
      mensaje += `<b>Horas estimadas</b> ${result.estimated} \n`;
      mensaje += `<b>Horas incurridas</b> ${result.actual} \n`;
      bot.sendMessage(opts.chat_id, mensaje, {
        parse_mode: "HTML",
      });
      bot.answerCallbackQuery(callbackQuery.id);
    }
  );
} else if (data.reporte) {
  if (data.reporte === "sms") {
    MenuService.getMeses().then((result) => {
      bot.sendMessage(opts.chat_id, `Elige un mes para el reporte`, {
        reply_markup: {
          inline_keyboard: result,
        },
        parse_mode: "HTML",
      });
    });
  } else {
    MenuService.getMesesE().then((result) => {
      bot.sendMessage(opts.chat_id, `Elige un mes para el reporte`, {
        reply_markup: {
          inline_keyboard: result,
```

```
        },
        parse_mode: "HTML",
      });
    });
  }
  bot.answerCallbackQuery(callbackQuery.id);
 } else if (data.mes) {
  var fechas = UtilsService.diasDeMeses(undefined, data.mes);
  var mensaje = `<b>Incurrido del mes ${data.mes}</b> \n`;
  var primero = fechas[0].fecha;
  var ultimo = fechas[fechas.length - 1].fecha;
  bot.sendMessage(id, ` <b>Consultando mes ${data.mes}...</b>`, {
    parse_mode: "HTML",
  });
  JiraService.getIssuesWorkLogForUserV2(id, primero, ultimo).then(
    (result) => {
      for (var f of fechas) {
        var existe = result.has(f.fecha);
        if (f.day === "S" || f.day === "D") {
          mensaje += `<code>${f.fecha} = </code> ${constantes.ICONO_FERIADO}
\n`;
        } else {
          if (existe) {
            var value = result.get(f.fecha);
            if (value.time < 32400) {
              mensaje += `<code>${f.fecha} = ${UtilsService.vonvertirHoras(
                value.time
              )}</code> (${value.sms}) ${constantes.ICONO_PENDIENT}\n`;
            } else {
              mensaje += `<code>${f.fecha} = ${UtilsService.vonvertirHoras(
                value.time
              )}</code> (${value.sms}) ${constantes.ICONO_OK}\n`;
            }
          } else {
            mensaje += `<code>${f.fecha} = 0h</code> ${constantes.ICONO_NO} \n`;
          }
        }
      }
      bot.sendMessage(opts.chat_id, mensaje, {
        parse_mode: "HTML",
      });
      bot.answerCallbackQuery(callbackQuery.id);
    }
  );
 } else if (data.mesE) {
  var fechas = UtilsService.diasDeMeses(undefined, data.mesE);
  var mensaje = `<b>Incurrido del mes ${data.mes}</b> \n`;
  var primero = fechas[0].fecha;
  var ultimo = fechas[fechas.length - 1].fecha;
  bot.sendMessage(id, ` <b>Consultando mes ${data.mesE}...</b>`, {
    parse_mode: "HTML",
  });
  const usuario = await CacheService.getCache(id);
  const dataResponse = await TelegramService.reporteIncurrido(
    bot,
```

```
      id,
      primero,
      ultimo,
      [usuario.name]
    );
    ExcelService.reportePorUsuario(
      dataResponse.data,
      data.mesE,
      dataResponse.fechas
    ).then((dataE) => {
      const buffer = Buffer.from(dataE);
      const fileOptions = {
        filename: "Reporte_Incurridos.xlsx",
        contentType:
          "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",
      };
      bot.sendDocument(
        id,
        buffer,
        {
          caption: `Incurrido mes ${data.mesE}`,
          parse_mode: "HTML",
        },
        fileOptions
      );
      bot.answerCallbackQuery(callbackQuery.id);
    });
    bot.answerCallbackQuery(callbackQuery.id);
  } else if (data.command === "deleteIssues") {
    const opts = {
      reply_markup: {
        inline_keyboard: [
          [
            {
              text: `Descargar Formato`,
              callback_data: JSON.stringify({
                deleteI: "dowFormato",
              }),
            },
            {
              text: `Subir Formato`,
              callback_data: JSON.stringify({
                deleteI: "upFormato",
              }),
            },
          ],
        ],
      },
      parse_mode: "HTML",
    };
    bot.sendMessage(
      id,
      "<b>Eliminar WorkLogs</b> Elige una de las opciones",
      opts
    );
```

```
    bot.answerCallbackQuery(callbackQuery.id);
  } else if (data.deleteI === "dowFormato") {
    bot
      .sendMessage(
        opts.chat_id,
        "Ingrese keys de las tareas. Ejm 'CBA-123,ABC-002'",
        {
          reply_markup: {
            force_reply: true,
          },
          parse_mode: "HTML",
        }
      )
      .then(async (response) => {
        const replyListenerId = bot.onReplyToMessage(
          response.chat.id,
          response.message_id,
          async (msg) => {
            bot.removeReplyListener(replyListenerId);
            const {
              chat: { id },
            } = msg;
            const keys = msg.text.split(",");
            if (keys.length !== 0) {
              const usuario = await CacheService.getCache(id);
              let dataWork = [];
              for(let key of keys){
                const worklogs = await
JiraService.getAllWorklogs(key,id,1,null,null,usuario.name);
                dataWork.push(...worklogs.map(x=>{return
{...x,issue:key,fecha:UtilsService.converterDateMiliseconds(x.started,x.updateAuthor
.timeZone)}}))
              }
              await dataWork.sort((a, b) => a.fecha - b.fecha);
              const dataResponse = await
TelegramService.dataExcelEliminarWorklog(dataWork);
              ExcelService.reporteEliminar(dataResponse,usuario.name).then((dataE)
=> {
                const buffer = Buffer.from(dataE);
                const fileOptions = {
                  filename: `Worklogs_${msg.text.replace(",","_")}.xlsx`,
                  contentType:
                    "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",
                };
                bot.sendDocument(
                  id,
                  buffer,
                  {
                    caption: `Worklogs ${msg.text} `,
                    parse_mode: "HTML",
                  },
                  fileOptions
                );
                bot.answerCallbackQuery(callbackQuery.id);
              });
```

```
          } else {
            bot.sendMessage(id, "<b>Error debe ingresar un key valido</b>");
          }
          console.log(keys);
        }
      );
    });
  bot.answerCallbackQuery(callbackQuery.id);
 } else if (data.deleteI === "upFormato") {
  bot.sendMessage(opts.chat_id, "sube el formato worklogs a eliminar", {
    reply_markup: {
      force_reply: true,
    },
  }).then(async (response) => {
    const replyListenerId = bot.onReplyToMessage(
      response.chat.id,
      response.message_id,
      async (msg) => {
        bot.removeReplyListener(replyListenerId);
        const {
          chat: { id },
        } = msg;
        if (msg.document === undefined || msg.document.mime_type !==
'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet'
        ) {
          bot.sendMessage(
            id,
            ` <b>formato incorrecto vuelva a intentar</b>`,
            {
              parse_mode: "HTML",
            }
          );
        } else {
          bot.sendMessage(id, ` <b>Procesando...</b>`, {
            parse_mode: "HTML",
          });
          TelegramService.getPathDocument(msg.document.file_id).then(
            (response) => {
              TelegramService.getDocument(response).then(async (res) => {
                const newBuffer = Buffer.from(res);
              let filas = await ExcelService.leerExcelEliminar(newBuffer, bot, id);
              filas = filas.filter(x => x.eliminar === "SI");
                for (var tarea of filas) {
                  const ID = tarea.id;
                  await JiraService.deleteWorklog(id,tarea.jirakey,tarea.id).then(
                    (res) => {
                      bot.sendMessage(
                        id,
                        ` <b>ID: ${tarea.id}</b> exitoso`,
                        {
                          parse_mode: "HTML",
                        }
                      );
                    },
                    (err) => {
```

```
                    bot.sendMessage(
                      id,
                      ` <b>ID: ${ID}</b> error`,
                      {
                        parse_mode: "HTML",
                      }
                    );
                  }
                );
              }
              bot.sendMessage(id, ` <b>Eliminado Masivo Finalizada</b>`, {
                parse_mode: "HTML",
              });
              bot.answerCallbackQuery(callbackQuery.id);
            });
          },
          (error) => {}
        );
      }

    }
  );
 });
 } else if(data.command === 'tutoriales'){
  bot.sendMessage(id, ` <b>Estamos trabajando, muy pronto tendra la opción disponible</b>`, {
    parse_mode: "HTML",
  });
  bot.answerCallbackQuery(callbackQuery.id);
 }else if(data.command === 'equipo'){
  const usuario = await CacheService.getCache(id);//usuario.rol,usuario.name
  EquipoService.findEquipoByRoleV2(usuario.rol,usuario.name).then(resulEq=>{
    for (var res of resulEq) {
      const opts = {
        reply_markup: {
          inline_keyboard: res.menu,
        },
        parse_mode: "HTML",
      };
      bot.sendMessage(id, res.mensaje, opts);
    }

  });
 }else if(data.idequipo){
  let idEquipo = data.idequipo;
  bot
  .sendMessage(opts.chat_id, "Ingrese rango de fecha fechaInicio,fechaFin ejm:(2024-01-01,2024-01-31)", {
    reply_markup: {
      force_reply: true,
    },
    parse_mode: "HTML",
  }).then(response =>{

    const replyListenerId = bot.onReplyToMessage(
```

```
      response.chat.id,
      response.message_id,
      async (msg) => {
        bot.removeReplyListener(replyListenerId);
        const dataCache = await CacheService.getCache(id);
        const mensaje = msg.text.split(",");
        const fechIni = mensaje[0];
        const fechFin = mensaje[1];
        const rango = `${fechIni} - ${fechFin}`;//data.rol,data.name,idEquipo

EquipoService.peoplesByUserRoleIdEquipo(dataCache.rol,dataCache.name,idEquip
o).then(async (peoplesResp)=>{
      await bot.sendMessage(id, ` <b>Consultando rango ${rango}...</b>`, {
          parse_mode: "HTML",
        });
        const dataResponse = await TelegramService.reporteIncurrido(
          bot,
          id,
          fechIni,
          fechFin,
          peoplesResp,
          dataCache
        );
        const rangoDes = `Incurrido del ${fechIni} al ${fechFin}`;
        ExcelService.reportePorUsuario(
          dataResponse.data,
          rangoDes,
          dataResponse.fechas
        ).then((dataE) => {
          const buffer = Buffer.from(dataE);
          const fileOptions = {
            filename: "Reporte_Incurridos.xlsx",
            contentType:
              "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",
          };
          bot.sendDocument(
            id,
            buffer,
            {
              caption: ` ${rangoDes}`,
              parse_mode: "HTML",
            },
            fileOptions
          );
        });
      });


      bot.answerCallbackQuery(callbackQuery.id);
    })
  })

 }else if(data.teamSms){
   let teamSms = data.teamSms;
   const fecha = UtilsService.convertirMomentString(UtilsService.fechaActual());
```

```
    var mensaje = `<b>Reporte del día</b> \n`;
    await bot.sendMessage(opts.chat_id, `Consultando fecha ${fecha}`, {
      parse_mode: "HTML",
    })
    const dataCache = await
CacheService.getCache(id);//data.rol,data.name,teamSms

EquipoService.peoplesByUserRoleIdEquipo(dataCache.rol,dataCache.name,teamS
ms).then(async (peoplesResp)=>{

      const dataResponse = await TelegramService.reporteIncurrido(
        bot,
        id,
        fecha,
        fecha,
        peoplesResp,
        dataCache
      );
      for (const reporte  of dataResponse.data) {
        const horaR = reporte[1].split("|")[0] !== 0 ? Number(reporte[1].split("|")[0])
:reporte[1].split("|")[0];
        const nameR =  reporte[0] ? reporte[0] : "";

          if (horaR === 0) {
            mensaje += `<b>${nameR} = 0h</b> ${constantes.ICONO_NO} \n`;

          } else if(horaR < 9){
            mensaje += `<b>${nameR} = ${horaR}h</b>
${constantes.ICONO_PENDIENT}\n`;
          } else {
            mensaje += `<b>${nameR} = ${horaR}h</b> ${constantes.ICONO_OK}\n`;
          }

      }
      bot.sendMessage(opts.chat_id, `<pre><code>${mensaje}</code></pre>`, {
        parse_mode: "HTML",
      });
    });

    bot.answerCallbackQuery(callbackQuery.id);
  }else if(data.teamNoI){
    let teamNoI = data.teamNoI;
    const fecha = UtilsService.convertirMomentString(UtilsService.fechaActual());
    var mensaje = `<b>Reporte del día</b> \n`;
    await bot.sendMessage(opts.chat_id, `Consultando fecha ${fecha}`, {
      parse_mode: "HTML",
    })
    const dataCache = await
CacheService.getCache(id);//data.rol,data.name,teamSms

EquipoService.peoplesByUserRoleIdEquipo(dataCache.rol,dataCache.name,teamN
oI).then(async (peoplesResp)=>{

      const idTelegram = 19111992; //id fijo del teleragm de dither
```

```
    const dataResponse = await TelegramService.reporteNoIncurridoV2(idTelegram,peoplesResp);
    ExcelService.reporteNoIncurridos(dataResponse.data[0]).then(async (dataE) => {
      const buffer = Buffer.from(dataE);

      const fileOptions = {
        filename: "NoIncurridos.xlsx",
        contentType:
          "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",
      };
      bot.sendDocument(
        id,
        buffer,
        {
          caption: `No Incurridos `,
          parse_mode: "HTML",
        },
        fileOptions
      );
      await EmailService.sendEmailFile(
        "jcisnerp@emeal.nttdata.com",
        '/Documentos/AUTOMATED/noincurrirTeam',
        "<b>FYI</b>",
        `teamNoI_${dataCache.name}_${teamNoI}`,
        buffer
      );
    });
  });

  bot.answerCallbackQuery(callbackQuery.id);
 }
 else if (data.command) {
  bot
    .sendMessage(opts.chat_id, "Ingrese tiempo a incurrir (1m,1h,1d)", {
      reply_markup: {
        force_reply: true,
      },
      parse_mode: "HTML",
    })
    .then(async (response) => {
      const replyListenerId = bot.onReplyToMessage(
        response.chat.id,
        response.message_id,
        async (msg) => {
          bot.removeReplyListener(replyListenerId);
          const {
            chat: { id },
          } = msg;
          bot
            .sendMessage(opts.chat_id, "Ingrese algun comentario", {
              reply_markup: {
                force_reply: true,
              },
              parse_mode: "HTML",
            })
```

```
            .then(async (responsecomentario) => {
              const replyListenerCometario = bot.onReplyToMessage(
                responsecomentario.chat.id,
                responsecomentario.message_id,
                async (msgcomentario) => {
                  bot.removeReplyListener(replyListenerId);
                  bot.sendMessage(id, ` <b>Incurriendo...</b>`, {
                    parse_mode: "HTML",
                  });

                  JiraService.addWorklog2(
                    msg.text,
                    id,
                    data.command,
                    msgcomentario.text,
                    null
                  )
                    .then((data) => {
                      JiraService.getIssues(msg, id).then((result) => {
                        bot.sendMessage(
                          id,
                          `Hola <b>${data.displayName}  </b> estan son tus tareas en jira`,
                          {
                            parse_mode: "HTML",
                          }
                        );
                        for (var res of result) {
                          bot.sendMessage(id, res.mensaje, {
                            reply_markup: {
                              inline_keyboard: res.menu,
                            },
                            parse_mode: "HTML",
                          });
                        }
                      });
                      bot.answerCallbackQuery(callbackQuery.id);
                    })
                    .catch((error) =>
                      bot.sendMessage(opts.chat_id, `<b>${error}</b>`, {
                        parse_mode: "HTML",
                      })
                    );
                }
              );
            });
        }
      );
    });
  }
});

const job = schedule.scheduleJob(constantes.LVcinco, async function (fireDate) {
  reporteScheduler();
});
```

```
//EVENTOS
bot.on("polling_error", console.log);

bot.on("document", async (msg) => {
  const {
    chat: { id },
  } = msg;
  if (msg.caption === "" || msg.caption !== "/people") {
    /*bot.sendMessage(id, ` <b>comando invalido</b>`, {
          parse_mode: 'HTML'
       });*/
  } else {
    if (
      msg.document === undefined ||
      msg.document.mime_type !==
        "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
    ) {
      bot.sendMessage(id, ` <b>formato incorrecto vuelva a intentar</b>`, {
        parse_mode: "HTML",
      });
    } else {
      bot.sendMessage(id, ` <b>Procesando...</b>`, {
        parse_mode: "HTML",
      });

      TelegramService.getPathDocument(msg.document.file_id).then(
        (response) => {
          TelegramService.getDocument(response).then((res) => {
            const newBuffer = Buffer.from(res);
            ExcelService.leerExcel(newBuffer, bot, id);
          });
        },
        (error) => {}
      );
    }
  }
});

bot.on("message", async (msg) => {
  const {
    chat: { id },
  } = msg;
  if (
    msg.text
      ?.toString()
      .toLowerCase()
      .indexOf(constantes.CONDICION_UPDATE_STATUS) === 0
  ) {
    const dataUser = await UsersService.getUser(id);
    let mensaje = msg.text.split(".");
    JiraService.updateStatus(mensaje[1], mensaje[2], id).then((result) => {
      bot.sendMessage(id, ` <b>Cambio de estado </b>  correctamente`, {
        parse_mode: "HTML",
      });
    });
```

```
  }
});

//COMANDOS

bot.onText(/csv/, async (msg) => {
  const mensaje = msg.text.replace("/csv ", "").split(",");
  const fechIni = mensaje[0];
  const fechFin = mensaje[1];
  let cabecera = "fecha(dd/mm/yyyy);hora(1m,1h,1d);issue key;comentarios\n";
  if (fechIni && fechFin) {
    const fechas = UtilsService.diasRangoFechasCsv(fechIni, fechFin);
    for (var dia of fechas) {
      if (dia.day !== "S" && dia.day !== "D") {
        cabecera += ` ${dia.fecha};;;;\n`;
      }
    }
  }
  const buff = Buffer.from(cabecera, "utf-8");
  const fileOptions = {
    filename: "formato.csv",
    contentType: "text/csv",
  };
  bot.sendDocument(
    msg.chat.id,
    buff,
    {
      caption: ` formato para la carga masiva`,
      parse_mode: "HTML",
    },
    fileOptions
  );
});

bot.onText(/dduahdua/, async (msg) => {
  var m = "";
  m = "<b>Condiciones climatologicas</b>" + "\n";
  m += "<b>Latitud: </b>" + "12" + "\n";
  m += "<b>Longitud: </b>" + "133" + "\n";
  m += " <b>bold</b>, <strong>bold</strong> \n";
  m += "<i>italic</i>, <em>italic</em> \n";
  m += " <u>underline</u>, <ins>underline</ins>\n";
  m +=
    "<s>strikethrough</s>,  <strike>strikethrough</strike>,  <del>strikethrough</del>
\n";
  m +=
    '<span class="tg-spoiler">spoiler</span>, <tg-spoiler>spoiler</tg-spoiler> \n';
  m +=
    '<b>bold <i>italic bold <s>italic bold strikethrough <span class="tg-spoiler">italic
bold strikethrough spoiler</span></s> <u>underline italic bold</u></i> bold</b> \n';
  m += '<a href="https://umane.everis.com/jiraito/browse/">inline URL</a> \n';
  m += '<a href="tg://user?id=123456789">inline mention of a user</a> \n';
  m += "<code>inline fixed-width code</code> \n";
  m += "<pre>pre-formatted fixed-width code block</pre> \n";
  m +=
```

```
   '<pre><code class="language-python">pre-formatted fixed-width code block written
in the Python programming language</code></pre> \n';
  const opts = {
    parse_mode: "HTML",
  };
  bot.sendMessage(msg.chat.id, m, opts);
});

bot.onText(/svg/, async (msg) => {
  const svgInit = `<svg xmlns:sodipodi="http://sodipodi.sourceforge.net/DTD/sodipodi-
0.dtd" xmlns="http://www.w3.org/2000/svg" xmlns:svg="http://www.w3.org/2000/svg"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cc="http://creativecommons.org/ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/" contentScriptType="text/ecmascript"
id="svg3582" version="1.1" width="71mm" height="30mm"
sodipodi:docname="ETIQUETA DUMMY2.svg" ><defs
id="defs50"/><sodipodi:namedview id="namedview48" pagecolor="#ffffff"
bordercolor="#666666" borderopacity="1.0" height="99.25629mm"
width="209mm"/><metadata id="metadata3587"><rdf:RDF><cc:Work
rdf:about=""><dc:format>image/svg+xml</dc:format><dc:type
rdf:resource="http://purl.org/dc/dcmitype/StillImage"/></cc:Work></rdf:RDF></metad
ata><text xml:space="preserve" style="font-style:normal;font-variant:normal;font-
weight:bold;font-stretch:normal;font-size:12.7433px;line-height:1.25;font-family:sans-
serif;font-variant-ligatures:normal;font-variant-caps:normal;font-variant-
numeric:normal;font-variant-east-asian:normal;letter-spacing:0px;word-
spacing:0px;fill:#000000;fill-opacity:1;stroke:none;stroke-width:0.955749"
x="56.311035" y="12.531939" id="text3750"
transform="matrix(1.025474,0.00151464,-0.00153723,0.97515654,0,0)"><tspan
id="tspan3748" x="56.311035" y="12.531939" style="font-style:normal;font-
variant:normal;font-weight:bold;font-stretch:normal;font-size:12.7433px;font-
family:sans-serif;font-variant-ligatures:normal;font-variant-caps:normal;font-variant-
numeric:normal;font-variant-east-asian:normal;stroke-
width:0.955749">%NRO_DOCUMENTO%</tspan></text><g id="g21127"
transform="translate(-27.419817,-106.46024)"><text xml:space="preserve"
style="font-style:normal;font-variant:normal;font-weight:normal;font-
stretch:normal;font-size:9px;line-height:1.25;font-family:sans-serif;font-variant-
ligatures:normal;font-variant-caps:normal;font-variant-numeric:normal;font-variant-
east-asian:normal;letter-spacing:0px;word-spacing:0px;fill:#000000;fill-
opacity:1;stroke:none;stroke-width:0.955749" x="100.7402" y="145.85164"
id="text3750-9" transform="matrix(1.0592244,0.00146638,-
0.00158782,0.9440848,0,0)"><tspan id="tspan3748-1" x="100.7402" y="145.85164"
style="font-style:normal;font-variant:normal;font-weight:normal;font-
stretch:normal;font-size:9px;font-family:sans-serif;font-variant-ligatures:normal;font-
variant-caps:normal;font-variant-numeric:normal;font-variant-east-
asian:normal;stroke-width:0.955749">%NOMBRES%</tspan></text><text
xml:space="preserve" style="font-style:normal;font-variant:normal;font-
weight:normal;font-stretch:normal;font-size:9px;line-height:1.25;font-family:sans-
serif;font-variant-ligatures:normal;font-variant-caps:normal;font-variant-
numeric:normal;font-variant-east-asian:normal;letter-spacing:0px;word-
spacing:0px;fill:#000000;fill-opacity:1;stroke:none;stroke-width:0.955749"
x="104.11533" y="158.93977" id="text3750-9-7"
transform="matrix(1.025474,0.00151464,-0.00153723,0.97515654,0,0)"><tspan
id="tspan3748-1-7" x="104.11533" y="158.93977" style="font-style:normal;font-
variant:normal;font-weight:normal;font-stretch:normal;font-size:9px;font-family:sans-
serif;font-variant-ligatures:normal;font-variant-caps:normal;font-variant-
```

```
numeric:normal;font-variant-east-asian:normal;stroke-
width:0.955749">%APELLIDOS%</tspan></text><text xml:space="preserve"
style="font-style:normal;font-variant:normal;font-weight:normal;font-
stretch:normal;font-size:7.5494px;line-height:1.25;font-family:sans-serif;font-variant-
ligatures:normal;font-variant-caps:normal;font-variant-numeric:normal;font-variant-
east-asian:normal;letter-spacing:0px;word-spacing:0px;fill:#000000;fill-
opacity:1;stroke:none;stroke-width:0.95575" x="124.2492" y="149.04515"
id="text3750-9-1" transform="matrix(0.86018972,0.00180568,-
0.00128946,1.1625315,0,0)"><tspan sodipodi:role="line" x="124.2492"
y="149.04515" style="stroke-width:0.95575"
id="tspan10035">%DIRECCION%</tspan><tspan sodipodi:role="line" x="124.2492"
y="158.4819" style="stroke-width:0.95575" id="tspan15656"/><tspan
sodipodi:role="line" id="tspan1148" x="124.2492" y="167.91866" style="stroke-
width:0.95575"/></text><text xml:space="preserve" style="font-style:normal;font-
variant:normal;font-weight:normal;font-stretch:normal;font-size:9px;line-
height:1.25;font-family:sans-serif;font-variant-ligatures:normal;font-variant-
caps:normal;font-variant-numeric:normal;font-variant-east-asian:normal;letter-
spacing:0px;word-spacing:0px;fill:#000000;fill-opacity:1;stroke:none;stroke-
width:0.955749" x="104.2017" y="216.54085" id="text3750-9-5"
transform="matrix(1.025474,0.00151464,-0.00153723,0.97515654,0,0)"><tspan
id="tspan3748-1-9" x="104.2017" y="216.54085" style="font-style:normal;font-
variant:normal;font-weight:normal;font-stretch:normal;font-size:9px;font-family:sans-
serif;font-variant-ligatures:normal;font-variant-caps:normal;font-variant-
numeric:normal;font-variant-east-asian:normal;stroke-
width:0.955749">%UBIGEO%</tspan></text><text xml:space="preserve"
style="font-style:normal;font-variant:normal;font-weight:bold;font-stretch:normal;font-
size:12px;line-height:1.25;font-family:sans-serif, Arial;font-variant-
ligatures:normal;font-variant-caps:normal;font-variant-numeric:normal;font-variant-
east-asian:normal;letter-spacing:0px;word-spacing:0px" x="27.684862"
y="137.00139" id="text6740"><tspan sodipodi:role="line" id="tspan6738"
x="27.684862" y="137.00139">NOMBRES: </tspan><tspan sodipodi:role="line"
x="27.684862" y="152.00139" id="tspan6742"/></text><text xml:space="preserve"
style="font-style:normal;font-variant:normal;font-weight:bold;font-stretch:normal;font-
size:12px;line-height:1.25;font-family:sans-serif, Arial;font-variant-
ligatures:normal;font-variant-caps:normal;font-variant-numeric:normal;font-variant-
east-asian:normal;letter-spacing:0px;word-spacing:0px" x="28.189165"
y="154.82602" id="text9238"><tspan sodipodi:role="line" id="tspan9236"
x="28.189165" y="154.82602">APELLIDOS:</tspan></text><text
xml:space="preserve" style="font-style:normal;font-variant:normal;font-
weight:bold;font-stretch:normal;font-size:12px;line-height:1.25;font-family:sans-serif,
Arial;font-variant-ligatures:normal;font-variant-caps:normal;font-variant-
numeric:normal;font-variant-east-asian:normal;letter-spacing:0px;word-spacing:0px"
x="27.685259" y="174.90692" id="text9682"><tspan sodipodi:role="line"
id="tspan9680" x="27.685259" y="174.90692">DIRECCIÓN:</tspan></text><text
xml:space="preserve" style="font-style:normal;font-variant:normal;font-
weight:bold;font-stretch:normal;font-size:12px;line-height:1.25;font-family:sans-serif,
Arial;font-variant-ligatures:normal;font-variant-caps:normal;font-variant-
numeric:normal;font-variant-east-asian:normal;letter-spacing:0px;word-spacing:0px"
x="27.76729" y="210.47606" id="text11006"><tspan sodipodi:role="line"
id="tspan11004" x="27.76729" y="210.47606">UBIGEO:</tspan></text></g></svg>`;
  let svg = "" + svgInit;
  const parameters = {
    NRO_DOCUMENTO: "70076364",
    NOMBRES: "julio cesar",
    APELLIDOS: "cisneros palomino",
```

```
    DIRECCION: "data.direccion",
    UBIGEO: "data.ubigeo",
  };

  for (var [key, value] of Object.entries(parameters)) {
    svg = svg.replace(`%${key}%`, value);
  }
  let s = await svg2png({
    input: svg,
    encoding: "buffer",
    format: "jpge",
  });
  // let buff = new Buffer(data.foto, 'base64');
  //bot.sendPhoto(id, buff );
  bot.sendPhoto(msg.chat.id, s);
});

bot.onText(/getLocation/, (msg) => {
  const opts = {
    reply_markup: JSON.stringify({
      keyboard: [
        [{ text: "Location", request_location: true }],
        [{ text: "Contact", request_contact: true }],
      ],
      resize_keyboard: true,
      one_time_keyboard: false,
    }),
  };
  bot.sendMessage(msg.chat.id, "Contact and Location request", opts);
});

bot.onText(/report/, async (msg) => {
  // formato a enviar /report fechIni,fechFin,usuarios
  //fechas en formato yyyy-mm-dd, usuarios separado con espacio ==> /report 2022-
01-01,2022-01-14, user1 user2 user3
  const id = msg.chat.id;
  const texto = msg.text.split(" ");
  if (texto[0] === "/report") {
    const data = await CacheService.getCache(id);
    const mensaje = msg.text.replace("/report ", "").split(",");
    const fechIni = mensaje[0];
    const fechFin = mensaje[1];
    const usuarios =
      mensaje[2] && data.isSupervisor
        ? mensaje[2].split(" ").filter((x) => x !== "")
        : [null];
    const rango = `${fechIni} - ${fechFin}`;

    bot.sendMessage(id, ` <b>Consultando rango ${rango}...</b>`, {
      parse_mode: "HTML",
    });
    const dataResponse = await TelegramService.reporteIncurrido(
      bot,
      id,
      fechIni,
```

```
      fechFin,
      usuarios,
      data
    );
    const rangoDes = `Incurrido del ${fechIni} al ${fechFin}`;
    ExcelService.reportePorUsuario(
      dataResponse.data,
      rangoDes,
      dataResponse.fechas
    ).then((dataE) => {
      const buffer = Buffer.from(dataE);
      const fileOptions = {
        filename: "Reporte_Incurridos.xlsx",
        contentType:
          "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",
      };
      bot.sendDocument(
        id,
        buffer,
        {
          caption: ` ${rangoDes}`,
          parse_mode: "HTML",
        },
        fileOptions
      );
    });
  } else {
    bot.sendMessage(id, `... `, {
      parse_mode: "HTML",
    });
  }
});

bot.onText(/persona/, async (msg) => {
  const id = msg.chat.id;
  const mensaje = msg.text.replace("/persona ", "").split(",");
  const usuario = mensaje[0];
  bot.sendMessage(id, ` <b>Consultando Persona ${usuario}...</b>`, {
    parse_mode: "HTML",
  });
  infoPersona(bot, id, usuario);
});

bot.onText(/clearCache/, async (msg) => {
  const id = msg.chat.id;
  const mensaje = msg.text.replace("/clearCache ", "").split(",");
  const key = mensaje[0];
  if (key) {
    CacheService.deleteCache(key);
  } else {
    CacheService.clearCache();
  }
  bot.sendMessage(id, ` <b>Cache liberado</b>`, {
    parse_mode: "HTML",
  });
```

```
});

bot.onText(/sendupdate/, async (msg) => {
  const id = msg.chat.id;
  await TelegramService.messageUpdateUsers(bot, id);
});

bot.onText(/noincurridos/, async (msg) => {
  const id = msg.chat.id;
  const idTelegram = 19111992; //id fijo del teleragm de dither
  const dataResponse = await TelegramService.reporteNoIncurrido(idTelegram);
  ExcelService.reporteNoIncurridos(dataResponse.data[0]).then(async (dataE) => {
    const buffer = Buffer.from(dataE);

    const fileOptions = {
      filename: "NoIncurridos.xlsx",
      contentType:
        "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",
    };
    bot.sendDocument(
      id,
      buffer,
      {
        caption: `Incurrido mes `,
        parse_mode: "HTML",
      },
      fileOptions
    );
    await EmailService.sendEmailFile(
      "jcisnerp@emeal.nttdata.com",
      "No incurridos",
      "<b>FYI</b>",
      `NoInCurridos`,
      buffer
    );
  });
});

bot.onText(/emailreport/, async (msg) => {
  const m = "Reporte terminado de generar";
  const idTelegram = msg.chat.id;
  const opts = {
    parse_mode: "HTML",
  };
  await reporteScheduler(bot, idTelegram);
  bot.sendMessage(msg.chat.id, m, opts);
});

bot.onText(/worklog/, async (msg) => {
  const id = msg.chat.id;
    const mensaje = msg.text.replace("/worklog ", "").split(",");
    const fecha = mensaje[0];
    const issueKey = mensaje[1];
    const time = mensaje[2];
    const comment = mensaje[3];
```

```
    const usuario = mensaje[4];

    bot.sendMessage(id, ` Incurriendo <b>${time}</b> en la tarea <b>${issueKey}</b>
en la siguiente fecha <b>${fecha}</b>`, {
      parse_mode: "HTML",
    });
    JiraService.addWorklog2(
      time,
      id,
      issueKey,
      comment,
      fecha,
      usuario
    ).then(
      (res) => {
        bot.sendMessage(
          id,
          ` <b>Fecha: ${fecha}</b> exitoso`,
          {
            parse_mode: "HTML",
          }
        );
      },
      (err) => {
        bot.sendMessage(
          id,
          ` <b>Fecha: ${fecha}</b> error`,
          {
            parse_mode: "HTML",
          }
        );
      }
    );

});

bot.onText(/help/, async (msg) => {
  var m = "";
  m += "<b>Comandos disponibles</b>" + "\n";
  m += "<b>IMPORTANTE: </b>" + "\n";
  m += "* El formato de las fechas es: <b>yyyy-mm-dd</b> ejemplo: 2023-01-25" +
"\n";
  m += "* <b>timeSpent</b>: tiempo a incurrir en formato de horas, ejm (1m,1h,1d)" +
"\n";
  m += "<b>COMANDOS: </b>" + "\n";
  m += "<code>/report fechaInicio,fechaFin</code> - comando que devuelve un excel
de los incurridos por el rango de fechas \n";
  m += "<code>/csv fechaInicio,fechaFin</code> - comando que devuelve un csv por
el rango de fechas, para la carga masiva \n";
  m += "<code>/worklog fechaIcurrido,issueKey,timeSpent,comment</code> -
comando que registra el worklog en una tarea\n";

  const opts = {
    parse_mode: "HTML",
  };
```

```
  bot.sendMessage(msg.chat.id, m, opts);
});

bot.onText(/yt/, async (msg) => {
const id = msg.chat.id;
   const mensaje = msg.text.replace("/yt ", "").split(",");
const url = mensaje[0];
   YoutubeService.getVideo(url).then((dataE) => {
     const buffer = Buffer.from(dataE);
     const fileOptions = {
       filename: "Reporte_Incurridos.mp4",
       contentType:
         "video/mp4",
     };
     bot.sendVideo(
       id,
       buffer,
       {
         caption: 'video',
         parse_mode: "HTML",
       },
       fileOptions
     );
   });
});


//FUNCIONES

async function reporteScheduler(botTE, idTelegram) {
  console.log("[START] Scheduler");
  const id = 19111992; //id fijo del teleragm de rob_mvasquep@emeal.nttdata.com
  const data = await CacheService.getCache(id);
  const listaEquipos = await EquipoService.listEquipos();
  const fechaIni = UtilsService.convertirMomentString(
    UtilsService.fechaActual().subtract(3, "d")
  );
  const fechaFIn = UtilsService.convertirMomentString(
    UtilsService.fechaActual()
  );
  for (const eq of listaEquipos) {
    console.log(`[EQUIP] ${eq.key}`);
    if (eq.emailsSendTeams && eq.emailsSendTeams !== "") {
      const dataResponse = await TelegramService.reporteIncurrido(
        bot,
        id,
        fechaIni,
        fechaFIn,
        eq.peoples,
        data,
        true
      );
      const rangoDes = `Incurrido del ${fechaIni} al ${fechaFIn}`;
    await  ExcelService.reportePorUsuario(
        dataResponse.data,
```

```
        rangoDes,
        dataResponse.fechas
      ).then(async (dataE) => {
        const buffer = Buffer.from(dataE);
        await EmailService.sendEmailFile(
          eq.emailsSendTeams,
          eq.pathOneDrive,
          "<b>FYI</b>",
          `IncurridoEq_${eq.key}`,
          buffer,
          botTE,
          idTelegram
        );
      });
    }
  }
  console.log("[STOP] Scheduler");
}

async function infoPersona(bot, id, user) {
  PeopleService.obtenerPersonaByUsuario(user, id).then(async (result) => {
    if (result) {
      var mensaje = `<b> Mi información Personal</b> \n`;
      mensaje += `\n`;
      mensaje += `🆔 <code>${result.idEmpleado} </code> \n`;
      mensaje += `<b>👤 ${result.nombres} ${result.apellidos} </b> \n`;
      mensaje += `<b>📂 ${result.categoria}</b> \n`;
      mensaje += `<b>${result.estado === "ACTIVE" ? "✅" : "❌"} ${
        result.estado
      }</b> \n`;
      mensaje += `<b>📅 ${result.fechaAlta}</b> \n`;
      mensaje += `<b>🧑‍💻 ${result.tecnologia}</b> \n`;
      mensaje += `<b> </b> \n`;
      mensaje += `<code>ASIGNACIONES</code>`;
      for (const asig of result.asignacion) {
        mensaje += `<b> </b> \n`;
        mensaje += `<b>Cliente: </b> ${asig.servicio}  \n`;
        mensaje += `<b>Equipo: </b> ${asig.servicioTeam}  \n`;
        mensaje += `<b>Fecha Inicio: </b> ${asig.fechaInicio} \n`;
        if (asig.fechaFin) {
          mensaje += `<b>Fecha Fin: </b> ${asig.fechaFin} \n`;
        }
        mensaje += `<b>Tecnología: </b> ${asig.tecnologia} \n`;
      }
      bot.sendMessage(id, mensaje, {
        parse_mode: "HTML",
      });
    } else {
      bot.sendMessage(id, `<b>Usuario no encontrado</b>`, {
        parse_mode: "HTML",
      });
    }
  });
}
```

Figura 38
CONTROLLER JOBS

```
'use strict'
require('dotenv').config()

const TelegramService = require('./services/telegram.service');
const ExcelService = require('./services/excel.service');
const EmailService = require('./services/email.service');
const schedule = require('node-schedule');
const TeamsService = require('./services/teams.service');



const job = schedule.scheduleJob("35 22 * * 1-5", async function (fireDate) {
    try{
        console.log("[START] NO INCURRIDOS");
        const idTelegram = 19111992; //id fijo del teleragm de dither
        const dataResponse = await TelegramService.reporteNoIncurrido(idTelegram);
        ExcelService.reporteNoIncurridos(dataResponse.data[0]).then(async (dataE) =>
{
            const buffer = Buffer.from(dataE)
           await                    EmailService.sendEmailFile('jcisnerp@emeal.nttdata.com',
'/incurridos/reportesTeams', '<b>FYI</b>',
                    `NoInCurridos`, buffer)
            console.log("[STOP] NO INCURRIDOS");
        })
    }catch(error){
        console.log(error);
    }finally{
        console.log('finally');
    }

});

/**
 const enviarMensajeTeams = schedule.scheduleJob("* 25 * * * *", async function
(fireDate) {
    try{
        console.log("[START] ENVIO TEAMS");
        await TeamsService.sendSMSTeams()
        console.log("[STOP] ENVIO TEAMS");
    }catch(error){
        console.log(error);
    }finally{
        console.log('finally');
    }
 TelegramService.autoIncurridosEquipos(1821658527);
});*/


 const generarExcelEnvioGrupoTeams = schedule.scheduleJob("35 21 * * 1-5", async
function (fireDate) {
    try{
        console.log("[START] REPORTE GRUPO TEAMS");
```

```
        const dataResponse = await TelegramService.reporteExcelGrupoTeams();
        ExcelService.reporteGrupoTeams(dataResponse.data[0]).then(async     (dataE)
=> {
            const buffer = Buffer.from(dataE)
           await                EmailService.sendEmailFile('jcisnerp@emeal.nttdata.com',
'/Documentos/AUTOMATED', '<b>FYI</b>',
                    `Reporte grupos teams`, buffer)
            console.log("[STOP] REPORTE GRUPO TEAMS");
        })
    }catch(error){
        console.log(error);
    }finally{
        console.log('finally');
    }

});
```

PRUEBAS DE INTEGRACIÓN

Tabla 53. Reporte de pruebas unitarias. Primera Iteración

HU	CLASE FUNCIONAL	CLASE PRUEBA UNITARIA	RESULTADO
1	users.service.js	users.service.test.js	Satisfactorio
2	Users.service.js	Users.service.test.js	Satisfactorio
3	Menu.service.js	Menu.service.test.js	Satisfactorio
4	Jira.service.js	Jira.service.test.js	Satisfactorio
5	Jira.service.js	Jira.service.test.js	Satisfactorio

Tabla 54. Reporte de pruebas de integración. Primera Iteración

HU	TI	CLASE FUNCIONAL	EVENTO	RESULTADO
1	1	Jira.service.js +users.service.js	Evento: /start + callback login	Satisfactorio
2	2	Jira.service.js Users.service.js	Evento: callback login + código	Satisfactorio
3	3	Menu.service.js	Evento: Texto hola o callback menú	Satisfactorio
4	4	Jira.service.js	Evento: callback tareas	Satisfactorio
5	5	Jira.service.js	Evento: callback proyectos	Satisfactorio

Tabla 55. Reporte de pruebas unitarias. Segunda Iteración

HU	CLASE FUNCIONAL	CLASE PRUEBA UNITARIA	RESULTADO
6	Jira.service.js	Jira.service.test.js	Satisfactorio

7	Jira.service.js	Jira.service.test.js	Satisfactorio
8	Jira.service.js	Jira.service.test.js	Satisfactorio
9	telegram.service.js	telegram.service.test.js	Satisfactorio
10	excel.service.js	excel.service.test.js	Satisfactorio

Tabla 56. Reporte de pruebas de integración. Segunda Iteración

HU	TI	CLASE FUNCIONAL	EVENTO	RESULTADO
6	6	Jira.service.js	Evento: callback command incurrir	Satisfactorio
7	7	Jira.service.js	Evento: callback cambiar estado	Satisfactorio
8	8	Jira.service.js	Evento: callback historial incurridos	Satisfactorio
9	9	telegram.service.js	Evento: callback reporte SMS	Satisfactorio
10	10	telegram.service.js + excel.service.js	Evento: callback reporte Excel	Satisfactorio

Tabla 57. Reporte de pruebas unitarias. Tercera Iteración

HU	CLASE FUNCIONAL	CLASE PRUEBA UNITARIA	RESULTADO
11	Jira.service.js	Jira.service.test.js	Satisfactorio
12	Jira.service.js	Jira.service.test.js	Satisfactorio
13	telegram.service.js	telegram.service.test.js	Satisfactorio
14	Jira.service.js	Jira.service.test.js	Satisfactorio

Tabla 58. Reporte de pruebas de integración. Tercera Iteración

HU	TI	CLASE FUNCIONAL	EVENTO	RESULTADO
11	11	telegram.service.js	Evento: comando /worklog	Satisfactorio
12	12	telegram.service.js	Evento: callback reporte múltiples usuarios	Satisfactorio
13	13	telegram.service.js	Evento: carga masiva incurridos	Satisfactorio
14	14	Jira.service.js	Evento: callback incurrir festivos	Satisfactorio

Tabla 59. Reporte de pruebas unitarias. Cuarta Iteración

HU	CLASE FUNCIONAL	CLASE PRUEBA UNITARIA	RESULTADO
15	Jira.service.js	Jira.service.test.js	Satisfactorio
16	Jira.service.js	Jira.service.test.js	Satisfactorio

17	telegram.service.js	telegram.service.test.js	Satisfactorio

Tabla 60. Reporte de pruebas de integración. Cuarta Iteración

HU	TI	CLASE FUNCIONAL	EVENTO	RESULTADO
15	15	Equipo.service.js + Jira.service.js	Evento: callback reporte equipo	Satisfactorio
16	16	telegram.service.js	Evento: alerta no incurridos	Satisfactorio
17	17	telegram.service.js	Evento: notificación automática diaria	Satisfactorio

PRUEBAS DE ACEPTACIÓN

Tabla 61. *Reporte de pruebas de aceptación. Autenticación en el bot.*

Nº CASO PRUEBA	1
Propósito	Iniciar sesión de usuarios en el bot de Telegram
ACTIVIDAD	
Inicialización	El usuario al ingresar enviara el comando /start y seguira el flujo de login
Descripción de datos de entrada	Usuario corto, código de verificación
RESULTADOS	
Esperado	Usuario autenticado correctamente y menú disponible
Reales	El usuario accede al menú del bot.

Tabla 62. *Reporte de pruebas de aceptación. Consultar mi información.*

Nº CASO PRUEBA	2
Propósito	Consultar información personal del usuario autenticado
ACTIVIDAD	
Inicialización	Usuario selecciona opción “Mi información”
Descripción de datos de entrada	Usuario autenticado
RESULTADOS	
Esperado	Se muestra la información personal del usuario
Reales	La información se presenta correctamente

Tabla 63. *Reporte de pruebas de aceptación. Actualizar información masiva.*

Nº CASO PRUEBA	3
Propósito	Permitir al usuario registrar incurridos de forma masiva
ACTIVIDAD	
Inicialización	Usuario carga archivo CSV a través del bot
Descripción de datos de	Archivo CSV con datos de fechas, horas y claves Jira

entrada	
RESULTADOS	
Esperado	Las entradas se registran correctamente en Jira
Reales	Todas las entradas del CSV se procesan sin errores

Tabla 64. *Reporte de pruebas de aceptación. Consultar información de otros usuarios.*

Nº CASO PRUEBA	**4**
Propósito	Consultar información de otros usuarios para supervisores
ACTIVIDAD	
Inicialización	Supervisor ingresa el comando /persona <user>
Descripción de datos de entrada	Usuario corto
RESULTADOS	
Esperado	Se muestra la información del usuario buscado
Reales	Los datos se visualizan correctamente

Tabla 65. *Reporte de pruebas de aceptación. Visualización de tareas.*

Nº CASO PRUEBA	**5**
Propósito	Permitir al usuario visualizar sus tareas asignadas
ACTIVIDAD	
Inicialización	Usuario selecciona opción “Mis tareas”
Descripción de datos de entrada	
RESULTADOS	
Esperado	Se listan las tareas actuales del usuario
Reales	Las tareas se muestran correctamente

Tabla 66. *Reporte de pruebas de aceptación. Registrar tiempo en tarea.*

Nº CASO PRUEBA	**6**
Propósito	Permitir al usuario registrar tiempo trabajado en una tarea
ACTIVIDAD	
Inicialización	Usuario selecciona tarea y registra tiempo con comentario
Descripción de datos de entrada	Tiempo, comentario
RESULTADOS	
Esperado	El tiempo se registra correctamente en Jira
Reales	Tiempo registrado exitosamente sin errores

Tabla 67. *Reporte de pruebas de aceptación. Cambiar estado de tarea.*

Nº CASO PRUEBA	**7**
Propósito	Permitir cambiar el estado de una tarea Jira
ACTIVIDAD	
Inicialización	Usuario selecciona opción cambio de estado
Descripción de datos de entrada	
RESULTADOS	
Esperado	El estado de la tarea se actualiza correctamente
Reales	Estado actualizado con éxito

Tabla 68. *Reporte de pruebas de aceptación. Ver historial de incurridos.*

Nº CASO PRUEBA	**8**
Propósito	Visualizar el historial de tiempo incurrido
ACTIVIDAD	
Inicialización	Usuario solicita el historial mediante opción worklogs
Descripción de datos de entrada	
RESULTADOS	
Esperado	Se muestra el historial de incurridos de la tarea
Reales	Historial visualizado correctamente.

Tabla 69. *Reporte de pruebas de aceptación. Reporte mensual (SMS).*

Nº CASO PRUEBA	**9**
Propósito	Generar reporte mensual de incurridos en formato SMS
ACTIVIDAD	
Inicialización	Usuario solicita el reporte desde el menú
Descripción de datos de entrada	Mes seleccionado
RESULTADOS	
Esperado	El reporte SMS se genera y muestra correctamente
Reales	Reporte generado sin errores.

Tabla 70. *Reporte de pruebas de aceptación. Reporte mensual (Excel).*

Nº CASO PRUEBA	**10**
Propósito	Generar reporte mensual de incurridos en formato Excel
ACTIVIDAD	
Inicialización	Usuario solicita el reporte desde el menú
Descripción de datos de entrada	Mes seleccionado
RESULTADOS	

Esperado	Archivo Excel generado con datos correctos
Reales	Archivo entregado y descargado sin inconvenientes

Tabla 71. *Reporte de pruebas de aceptación. Reporte por comando /worklog.*

Nº CASO PRUEBA	**11**
Propósito	Generar un reporte de incurridos mediante el comando /worklog
ACTIVIDAD	
Inicialización	Usuario ejecuta el comando /worklog con parámetros de fecha
Descripción de datos de entrada	FechaInicio, FechaFin
RESULTADOS	
Esperado	Se genera y entrega el reporte en Excel con registros correctos
Reales	Reporte generado y descargado sin errores

Tabla 72. *Reporte de pruebas de aceptación. Reporte por usuarios múltiples.*

Nº CASO PRUEBA	**12**
Propósito	Permitir generar reportes de incurridos para varios usuarios
ACTIVIDAD	
Inicialización	Supervisor usa el comando /worklog con varios usuarios
Descripción de datos de entrada	FechaInicio, FechaFin, Lista de usuarios
RESULTADOS	
Esperado	Se genera el reporte en Excel con registros por usuario
Reales	Reporte generado y descargado sin errores

Tabla 73. *Reporte de pruebas de aceptación. Registrar incurrido masivo.*

Nº CASO PRUEBA	**13**
Propósito	Registrar múltiples incurridos mediante un archivo CSV
ACTIVIDAD	
Inicialización	Usuario sube archivo CSV con datos de incurrido
Descripción de datos de entrada	CSV con fecha, hora, JiraKey y comentarios
RESULTADOS	
Esperado	Cada incurrido es registrado correctamente en Jira
Reales	Incurridos cargados exitosamente y confirmación mostrada

Tabla 74. *Reporte de pruebas de aceptación. Incurrir días festivos.*

Nº CASO PRUEBA	**14**
Propósito	Registrar automáticamente horas en días festivos
ACTIVIDAD	
Inicialización	Usuario solicita incurrir todos los festivos del año

Descripción de datos de entrada	Jira Key
RESULTADOS	
Esperado	Las horas se registran en todos los festivos sin errores
Reales	Registro masivo completado correctamente

Tabla 75. *Reporte de pruebas de aceptación. Reportes del equipo (lider).*

Nº CASO PRUEBA	**15**
Propósito	Generar reporte de incurridos de los miembros del equipo
ACTIVIDAD	
Inicialización	Jefe de equipo accede a la opción “Equipo” y selecciona rango de fechas
Descripción de datos de entrada	FechaInicio, FechaFin
RESULTADOS	
Esperado	El sistema genera y entrega el Excel con el detalle de incurridos del equipo
Reales	Reporte generado y descargado correctamente

Tabla 76. *Reporte de pruebas de aceptación. Alerta de no incurridos(lider).*

Nº CASO PRUEBA	**16**
Propósito	Enviar alertas a miembros que no han registrado incurridos
ACTIVIDAD	
Inicialización	Jefe de equipo accede a la opción “No incurridos”
Descripción de datos de entrada	Selección del equipo
RESULTADOS	
Esperado	Los miembros sin incurridos reciben alerta vía Teams y se genera un reporte
Reales	Alertas enviadas y reporte generado correctamente

Tabla 77. *Reporte de pruebas de aceptación. Notificación automática diaria.*

Nº CASO PRUEBA	**17**
Propósito	Ejecutar automáticamente revisión diaria de incurridos
ACTIVIDAD	
Inicialización	Scheduler ejecuta a las 5:30 am cada día
Descripción de datos de entrada	
RESULTADOS	
Esperado	Los usuarios que no han incurrido son notificados por Teams y se envían reportes a los líderes
Reales	Notificaciones y reportes enviados automáticamente

42.2 DISCUSIÓN

Allen (2001) y Vanderkam (2010) afirman que un registro estructurado del tiempo contribuye significativamente a la productividad individual y organizacional. En esta investigación se comprobó que, al facilitar el registro mediante un canal accesible como Telegram, los colaboradores pudieron incurrir sus worklogs de manera más oportuna, incluso fuera del entorno corporativo. Esta solución representa una aplicación práctica de los principios teóricos mencionados, al mejorar la trazabilidad de las tareas y permitir una toma de decisiones más informada por parte de los responsables de equipo.

El desarrollo de un bot en Telegram para el registro de worklogs en Jira permitió abordar eficazmente la problemática detectada en GDN PERU NTTDATA: la limitada accesibilidad para el registro de avances diarios desde dispositivos no corporativos. Mediante la metodología de desarrollo ágil XP, se diseñó, implementó y validó un prototipo funcional que optimiza tanto el tiempo como la eficiencia en el proceso de registro de actividades laborales.

CAPÍTULO V
CONCLUSIONES

43.1 CONCLUSIONES

a. El desarrollo e implementación de un bot en Telegram permitió automatizar el proceso de registro de worklogs en Jira, brindando a los colaboradores de GDN PERU NTTDATA una herramienta accesible. Esta solución resolvió la limitación de registrar únicamente desde equipos corporativos, facilitando el cumplimiento del registro diario incluso en escenarios remotos.

b. Se implementaron funcionalidades orientadas a la generación de reportes tanto individuales como grupales, incluyendo exportaciones en formato Excel, visualización de tareas y alertas automatizadas. Estas herramientas mejoraron la trazabilidad del trabajo y proporcionaron a los líderes de equipo medios efectivos de supervisión y control.

c. La solución propuesta no solo resolvió el problema planteado en el contexto de GDN PERU, sino que también demostró potencial de escalabilidad a otras sedes de NTTDATA o a organizaciones que utilicen plataformas similares a Jira. El enfoque modular y la integración con servicios como Microsoft Teams y Outlook fortalecen su aplicabilidad en entornos corporativos diversos.

CAPÍTULO VI
RECOMENDACIONES

44.1 RECOMENDACIONES

a. Integrar el bot con otras herramientas corporativas como Microsoft Power BI o Azure DevOps, lo que permitiría generar reportes más avanzados y mejorar la toma de decisiones basada en datos.

b. Monitorear periódicamente el rendimiento del bot y la frecuencia de uso a través de métricas definidas (número de registros, tiempo promedio, tasa de omisión), con el objetivo de realizar mejoras continuas sobre la base del feedback de los usuarios.

c. Evaluar la ampliación del uso del bot a otras áreas y equipos de GDN PERU NTTDATA, con el fin de estandarizar el proceso de registro de worklogs y asegurar la trazabilidad de las tareas en toda la organización.

44.1.1 Referencias Bibliográficas

Cameo, C. (2019). Bot de Telegram para Identificación de Farolas: https://oa.upm.es/63637/1/TFM_CARLOS_CAMEO_GILABERT.pdf

Gonzáles, E y Sánchez, P. (2017). BotMentor Bot de ayuda al estudiante en la plataforma telegra: https://docta.ucm.es/rest/api/core/bitstreams/2eaabbc5-b374-4f5b-9e38-1f8de5cd455c/content

Garibay, F. (2020). "Diseño e Implementacion de un aisistente virtual (chatbot) para ofrecer atención a los clientes de una aerolinea mexicana por medio de sus canales conversaionales": https://infotec.repositorioinstitucional.mx/jspui/bitstream/1027/402/1/INFOTEC_MGITIC_FAGO_27082020.pdf.

Ulrich, D. (2009). "HR Transformation: Building Human Resources from the Outside In". McGraw-Hill.

Teodoro, N., & Nieto, E. (2018). TIPOS DE INVESTIGACIÓN. http://repositorio.usdg.edu.pe/handle/USDG/34

Drucker, P.F. (1974). Management: Task, responsibilities, practices. Harper & Row.

Taylor, F.W. (1911). The principles of scientific management. Harper & Brothers.

Vanderkam, L. (2010), 168 hours: You have more time than you think. Portafolio.

Allen, D. (2001). Getting things done: The art of stress-free productivity. Penguin Books.

Covey, S. R. (1989). The 7 habits of highly effective people. Simon & Schuster.

Weitzenfeld, A. (s.f.). Ingeniería de software orientada a objetos: con UML, Java e internet. Madrid, España: Thomson.

Osorio, F. (2008). Base de datos relacionales: Teoría y práctica (1ª ed.). Madrid, España: Thomson.

Stair, R. y Reynolds, G. (1999). Principios de Sistemas de Información: Enfoque administrativo (4ª ed.). Madrid, España: Thomson.

Craing, I. (2002). The Interpretation of object-oriented programming languages (2ª ad.). Gran Bretaña: Springer-Verlag London.

Sipser, M. (1996). Introduction to the Theory of Computation. PWS Publishing Company.

Bratko, I. (2000). Prolog Programming for Artificial Intelligence. Pearson Education.

Colección Esencial (2011). Esencial Internet Explorer 9. Cataluña, España: Editions ENI.

Luján, S. (2001). Programación en internet: Clientes web. Alicante, España: Club Universitario.

Cafassi, E. (1998). Internet: Políticas y comunicaciones. Buenos Aires, Argentina: Biblos.

Redmond, E., y Wilson, J. R. (2012). Seven databases in seven weeks: A guide to modern databases and the NoSQL movement. Pragmatic Bookshelf.

Chodorow, K., & Dirolf, M. (2013). MongoDB: The definitive guide (2nd ed.). O'Reilly Media.

Wikipedia. (s.f.). Telegram Bot API. Wikipedia. Recuperado el 18 de febrero 2024, de https://es.wikipedia.org/wiki/Telegram_Bot_API#cite_note-

tbot-2

Barnes, D., & Kolling, M. (2007). Programación orientada a objetos con Java (Tercera ed.). (B. I. Brenta, Trad.) Madrid: pearson educación, s.a.

Díaz Labrador, M., & Collazo Garcia, A. (2013). La Programación Extrema.

Jeffries, R., Anderson, K., Hendrickson, Chet (2000). Extreme Programming Installed. Boston, USA: Addison-Wesley.

Joyanes, L. (1998). Programación orientada a objetos (2ª Ed.). España: Mcgraw W-Hill/Interamericana De España.

Kurose, J. y Ross, K. (2010). Redes de computadores: un enfoque descendente (5ª Ed.). Madrid, España: Pearson Educación S.A.

Letelier, P., Penadés, C. (2006). Métodologias ágiles para el desarrollo de software: Extreme Programming XP. Universidad Politécnica de Valencia. Valencia. España.

Luján, S. (2002). Programación de aplicaciones web: historia, principios básicos y clientes web. Madrid, España: Club Universitario.

Luque, I., Gómez, M., López, N. y Cerruela, G. (2002). Base de Datos. Desde Chen hasta Codd con Oracle. ORACLE. México, D.F., México: Alfaomega Grupo Editor.

Porras, E. (2010). Comparación de los procesos de desarrollo de software usando los métodos ICONIX y XP, caso: Comercializacion de la tara en la region de Ayacucho. Lima, Peru.

44.1.2 Lista de Abreviaturas

API: Application Programming Interface (Interfaz de Programación de Aplicaciones).
CSV: Comma-Separated Values (Valores Separados por Comas).
CIE: Código de Clasificación de Procedimientos.
GDN: Global Delivery Network (Red Global de Entrega).
HU: Historia de Usuario.
ID: Identifier (Identificador).
NTTDATA: Nippon Telegraph and Telephone Data Corporation.
SMS: Short Message Service (Servicio de Mensajería Corta).
TI: Tarea de Ingeniería.
XP: Extreme Programming (Programación Extrema).

44.1.3 Glosario

Análisis de requerimientos: Proceso de recopilación y definición de las funcionalidades necesarias que debe cumplir un sistema, basado en las necesidades del usuario y los objetivos del proyecto.

Bot: Programa automatizado diseñado para ejecutar tareas específicas mediante comandos o interacciones, como el registro de datos en una aplicación.

Desarrollo ágil: Enfoque de programación centrado en entregas incrementales, adaptabilidad al cambio y colaboración constante con el usuario final.

Historia de usuario (HU): Descripción breve de una funcionalidad desde la perspectiva del usuario, utilizada en metodologías ágiles para definir requerimientos.

Jira: Herramienta de gestión de proyectos y seguimiento de incidencias desarrollada por Atlassian, ampliamente utilizada para el control de tareas en entornos de desarrollo ágil.

MongoDB: Sistema de base de datos NoSQL orientado a documentos, que almacena la información en estructuras tipo JSON.

Notificación automática: Mensaje generado y enviado por el sistema sin intervención del usuario, usado para alertar o informar sobre eventos específicos.

Power Automate: Herramienta de automatización de flujos de trabajo desarrollada por Microsoft, que permite conectar aplicaciones y servicios para sincronizar datos o activar procesos.

Registro de worklogs: Proceso de ingreso del tiempo trabajado por un colaborador en una tarea específica, utilizado para el control de productividad.

Telegram: Aplicación de mensajería instantánea que permite la integración de bots para automatizar tareas y comunicaciones.

Visual Studio Code: Editor de código fuente multiplataforma desarrollado por Microsoft, utilizado para la programación de aplicaciones y scripts.

44.1.4 Anexos

44.1.4.1 Anexo 1. Matriz de consistencia

Título: BOT en Telegram para registro de worklogs en jira en GND PERU NTTDA, 2024.

PROBLEMAS	OBJETIVOS	VARIABLES	MÉTODO DE INVESTIGACIÓN
PROBLEMA GENERAL ¿De qué manera realizar registro de worklogs en jira de forma más eficiente en GDN PERU NTTDATA, 2024? **PROBLEMAS ESPECÍFICOS** a. ¿De qué manera gestionar los tiempos de registro de worklogs en jira? b. ¿Cómo realizar reportes de registro de worklogs en jira?	**OBJETIVO GENERAL** Desarrollar un bot en telegram para realizar registro de worklogs en jira de forma más eficiente en GDN PERU NTTDATA, 2024 **OBJETIVOS ESPECÍFICOS** a. Gestionar el tiempo de registro de worklogs en jira. b. Generar reportes de registro de worklogs en jira.	**VARIABLE INTERÉS** **X:** Registro de Worklogs **VARIABLES DESCRIPTIVAS** **X1:** Tiempo de registro **X2:** Reporte de registro	**TIPO DE INVESTIGACIÓN** Aplicada **NIVEL DE INVESTIGACIÓN** Descriptivo **DISEÑO DE INVESTIGACIÓN** No experimental **POBLACIÓN** Está conformada por 49 proyectos de GND Perú nttdata, 2024 **MUESTRA** Se realizará un muestreo no probabilístico, por conveniencia, siendo la muestra de 3 proyectos. **TÉCNICA** Análisis Documental **INSTRUMENTO** Registro de Análisis documental

44.1.4.2 **Anexo 2.** Instrumento para la recoleccion de datos para la varible de interes Registro Worklogs

El presente instrumento está diseñado para recopilar datos documentales relacionados con la variable principal 'Registro de worklog' y sus variables descriptivas: Tiempo de registro y Reporte de registro. Se basa en preguntas orientadoras que permiten validar el cumplimiento de requerimientos funcionales del sistema.

Variable: Registro worklog	**Pregunta**
Tiempo de registro: Minutos y Horas	¿Es necesario el procedimiento de autenticación de usuario?
	¿Es necesaria una visualización de las tareas asignadas?
	¿Es necesaria la visualización del estado de tareas?
	¿El bot permite registrar el tiempo de forma flexible (horas, minutos)?
	¿Es relevante cambiar el estado de una tarea desde el mismo entorno del bot?
	¿La carga masiva en CSV reduce tiempos operativos?
	¿Es necesario permitir el registro automático en feriados para ciertas tareas?
Reporte de Registro: sms y csv	¿Qué formato de reporte sería útil para visualizar los worklogs (mensaje, Excel, etc.)?
	¿Es relevante mostrar rangos de fechas o acumulados por día en los reportes?
	¿Es necesario alertar a los usuarios cuando no incurren en sus tareas?
	¿El líder necesita informes consolidados para monitoreo?
	¿El historial de incurridos facilita la trazabilidad de los registros anteriores?
	¿Se necesita consultar el historial de otros usuarios para gestión de equipos?
	¿El uso de comandos mejora la eficiencia del acceso a reportes personalizados?
	¿El líder de equipo necesita ver registros grupales?

UNSCH FACULTAD DE INGENIERÍA DE MINAS, GEOLOGÍA Y CIVIL

ACTA DE SUSTENTACIÓN DE TESIS N° 033-2025-FIMGC

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS

En la Universidad Nacional de San Cristóbal de Huamanga de la ciudad de Ayacucho, en cumplimiento a la **RESOLUCIÓN DECANAL No 252-2025-FIMGC-D**, a los **veintiocho días del mes de agosto 2025**, siendo las **4:00 p.m.**, reunidos en el **Auditorio de la Escuela Profesional de Ingeniería de Minas**, bajo la presidencia del **MSc. Ing. José Ernesto Estrada Cárdenas** y los miembros: **Mtro. Juan Carlos CARREÑO GAMARRA, Mtra. Celia Edith MARTINEZ CÓRDOVA** y **Mtra. Elinar CARRILLO RIVEROS,** actuando como secretario docente el **MSc. Ing. Saul Walter RETAMOZO FERNÁNDEZ**, para proceder a la sustentación de tesis para optar el **Título Profesional de Ingeniero de Sistemas**, del bachiller:

JULIO CESAR CISNEROS PALOMINO

Quien presentó la tesis denominada:

Bot en telegram para registro de worklogs en Jira en GDN Perú NTT Data, 2024

Los señores miembros del jurado, luego de expuesta la tesis y absueltas las preguntas, deliberaron y declararon:

Aprobado con diecisiete (17)

Siendo las **5:30 p.m.** del día 28 **de agosto de 2025**, culmina el acto de sustentación de tesis, y en conformidad con lo actuado, los miembros del jurado firman al pie del presente.

MSc. Ing. José Ernesto Estrada Cárdenas
Presidente

Mtro. Juan Carlos CARREÑO GAMARRA
Miembro

Mtra. Celia Edith MARTINEZ CÓRDOVA
Miembro

Mtra. Elinar CARRILLO RIVEROS
Miembro - Asesor

MSc. Saul Walter RETAMOZO FERNANDEZ
Secretario docente de la FIMGC

FACULTAD DE INGENIERÍA
DE MINAS Y CIVIL
Av. Independencia S/N
Ciudad Universitaria
Central Tel. 066 312510
Anexo 151

UNSCH FACULTAD DE INGENIERÍA DE MINAS, GEOLOGIA Y CIVIL

turnitin

CONSTANCIA DE ORIGINALIDAD DE TRABAJO DE INVESTIGACIÓN

CONSTANCIA Nº 023-2025-KPS-FIMGC/UNSCH

El que suscribe; responsable verificador de originalidad de trabajos de tesis de pregrado con el software Turnitin, en segunda instancia para las **Escuelas Profesionales** de la **Facultad de Ingeniería de Minas, Geología y Civil**; en cumplimiento a la **Resolución de Consejo Universitario Nº 039-2021-UNSCH-CU**, Reglamento de Originalidad de Trabajos de Investigación de la Universidad Nacional San Cristóbal de Huamanga y **Resolución Decanal Nº 697-2024-FIMGC-D**, deja constancia de originalidad de trabajo de investigación, que el/la Sr./Srta.

Nombres y Apellidos	:	Julio Cesar Cisneros Palomino
Escuela Profesional	:	INGENIERÍA DE SISTEMAS
Título de la Tesis	:	Bot en telegram para registro de worklogs en Jira en GDN Perú NTT Data, 2024
Evaluación de la Originalidad	:	**0%** Índice de Similitud
Identificador de la entrega	:	2742987125

Por tanto, según los Artículos 12, 13 y 17 del Reglamento de Originalidad de Trabajos de Investigación, es **PROCEDENTE** otorgar la **Constancia de Originalidad** para los fines que crea conveniente.

En señal de conformidad y verificación se firma la presente constancia

Ayacucho, 15 de setiembre de 2025

FIRMA DIGITAL

Firmado digitalmente por:
PERALTA SOTOMAYOR Karel
FAU 20143660754 soft
Motivo: Soy el autor del documento
Fecha: 15/09/2025 12:33:26-0500

Con depósito para Sustentación y Tramites
C.c. Archivo

FACULTAD DE INGENIERIA DE MINAS, GEOLOGIA Y CIVIL
Av. Independencia S/N Ciudad Universitaria
Central Tel. 066 312510
Anexo 151

Bot en telegram para registro de worklogs en Jira en GDN Perú NTT Data, 2024

por Julio Cesar Cisneros Palomino

Fecha de entrega: 05-sept-2025 04:47p. m. (UTC-0500)

Identificador de la entrega: 2742987125

Nombre del archivo: MEMORANDO_N_476-2025-CERTIFICADO_DE_ORIGINALIDAD-_JULIO_CESAR_CISNEROS_PALOMINO.pdf (3.95M)

Total de palabras: 28622

Total de caracteres: 183318

Bot en telegram para registro de worklogs en Jira en GDN Perú NTT Data, 2024

INFORME DE ORIGINALIDAD

0%	0%	0%	0%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

Excluir citas	Activo	Excluir coincidencias	< 30%
Excluir bibliografía	Activo		