UNIVERSIDAD DE SANCTI SPÍRITUS
José Martí Pérez

UNIVERSIDAD DE SANCTI SPÍRITUS “JOSÉ MARTÍ PÉREZ”

FACULTAD DE CIENCIAS TÉCNICAS Y ECONÓMICAS

CARRERA INGENIERÍA INFORMÁTICA

MIGRACIÓN DEL SOFTWARE HEREDADO SISTEMA INTEGRAL INFORMÁTICO DE LA DIRECCIÓN PROVINCIAL DE INFORMÁTICA DE SALUD PÚBLICA EN SANCTI SPÍRITUS

Trabajo de diploma en opción al título de Ingeniero Informático

Autor: Franco Armando Salgado León

Tutor: Ing. Julio Companioni Martínez

Sancti Spíritus

Diciembre 2022

“Año 64 de la Revolución”

PENSAMIENTO

Un programador tiene más oportunidades que nadie de innovar y emprender en este mundo tecnológico porque conoce las herramientas y ladrillos con los que se está creando esta nueva arquitectura económica mundial.

Rafael Gómez Blanes

AGRADECIMIENTOS

A mis padres, Leida y Armando, a quienes le debo todo por el apoyo incondicional que han tenido durante todo este recorrido desafiante.

A mi esposa Nancy quien ha estado siempre a mi lado en cada momento que lo necesitaba.

A mis hermanas Rachel y Lorraine, siendo esta última mi modelo a seguir académicamente.

A mis amistades, en especial Raúl y Alejo por estar ahí siempre que hacía falta un trago de cualquier bebida alcohólica para despejar.

Por supuesto a Osliani también, mi amistad más longeva con quien siempre he podido contar a través de los años.

A todo mi claustro de profesores en especial a Yunet y Vladimir por velar por mis pasos en esta última etapa del recorrido.

Por último, pero no menos importante a todos mis compañeros de curso, porque hemos llegado hasta el final por la unión y el apoyo que hemos tenido siempre.

¡GRACIAS!

RESUMEN

Los sistemas integrales de información desde sus orígenes han revolucionado la forma en la que se maneja la información dentro de las empresas, añadiendo organización y seguridad a estas sin necesidad de mucho personal para realizar estas labores. La Dirección Provincial de Informática en Sancti Spiritus contaba con este tipo de sistemas para gestionar la información dentro de su organización, en particular el Sistema Integral Informático; llamado así porque está enfocado en la administración desde el departamento de informática de las instituciones. El paso de los años y el avance de las tecnologías rodeaban a este sistema que no contaba ya con soporte ni actualizaciones quedando en un estado obsoleto en comparación con su entorno convirtiéndose en una isla de información dentro de la organización. Este proyecto se encargará de volver operativo al Sistema Integral Informático mediante una migración de software heredado utilizando como metodología a MADIISH, especializada en este tipo de migraciones. Será desarrollado con los frameworks Django y Nuxt.js para la parte lógica y visual respectivamente.

ABSTRACT

Integral information systems from their origins have revolutionized the way in which information is handled within companies, adding organization and security to them without the need for a lot of personnel to carry out these tasks. The Provincial Informatics Directorate in Sancti Spiritus had this type of system to manage information within its organization, particularly the Integral Informatics System; So called because it is focused on the administration from the Informatics department of the institutions. The passing of the years and the advancement of technologies surrounded this system, which no longer had support or updates, remaining in an obsolete state compared to its environment, becoming an island of information within the organization. This project will be responsible for making the Integral Informatics System operational through a legacy software migration using MADIISH as a methodology, specialized in this type of migration. It will be developed with the Django and Nuxt.js frameworks for the logical and visual part respectively.

Contenido

Índice de Ilustraciones

Índice de Tablas

INTRODUCCIÓN

En el año 1989 el periodista Erik Larson utilizó por primera vez el término Big Data en un artículo sobre el marketing y cómo se usarán los datos de los clientes, en los términos que actualmente conocemos. A partir de ahí se le acuñó el término Big Data al gran volumen de datos, tanto estructurados como no estructurados que inundan los negocios cada día.(Viana, 2015)

Los datos se convierten en información valiosa cuando hay un contexto inteligible detrás de él, y el conocimiento de ese contexto se genera a través de la experiencia, el análisis y la informática. Las empresas utilizan un sistema de información para recopilar y codificar datos que se analizan y extraen para mejorar el conocimiento operacional. Debido a que el conocimiento es poder, las empresas pueden utilizar conocimientos basados en sistemas de información para mejorar la toma de decisiones y optimizar los procesos de negocio.(Aguilar, 2015)

Los Sistemas Integrales de Información nacen de la necesidad de unificar la información dispersa en la organización. Años atrás y aun hoy las organizaciones contaban con sistemas informáticos para cada una de sus áreas. Estos sistemas conocidos también como islas de información no permitían el flujo eficaz de la información dentro de las áreas de la organización ocasionando duplicidad y desactualización de la información.(Cadre, 2017)

El Sistema Empresarial Cubano se encontraba inmerso en el proceso de adecuación del Modelo Económico de la Revolución Cubana. El funcionamiento y desarrollo de toda empresa cubana pasaba por la realidad de nuestra economía, resultado del bloqueo económico y comercial que por más de 60 años se ha ejercido sobre el país. La crisis económica global debilitaba además la salud de la empresa cubana en todas las ramas de la economía. (Lugones, 2016)

El desarrollo de los Sistemas de Información en las Empresas Cubanas era de carácter académico, es decir, se encontraba fundamentalmente en las universidades y no estaba totalmente desarrollado en el grueso de las empresas importantes de la economía cubana careciendo de presencia en la proyección de la gran mayoría de los empresarios. (Lugones, 2016)

La Dirección Provincial de Informática de Salud Pública en Sancti Spíritus se incorporó al desarrollo de Sistemas Integrales de información en la pasada década y en el año 2010 Juan Antonio Gonzales Herrera comenzó a conceptualizar la idea del Sistema Integral

Informático, un sistema integral de información con grandes aspiraciones que pretendía informatizar buena parte de los procesos que actualmente se gestionaban de forma manual, siento al comienzo un sistema pequeño pero modificable al poder agregar y eliminar módulos de lo que fuese necesario. Su desarrollo comenzó en el 2014 y para finales de 2015 su puesta en marcha. El resultado final fue un sistema disponible para todos los informáticos de las unidades de salud de la provincia, una aplicación web que contaba con una serie de módulos que permitían automatizar procesos tales como inspección de departamentos, mensajería, recursos humanos, planificación, entre otros; haciendo más sencillo el trabajo de los trabajadores de la entidad y contaba con una interfaz basada en kingAdmin, una plantilla de bootstrap 4 que facilitaba la interacción del usuario con el sistema.

Este sistema está implementado sobre un framework. Estos se actualizan con regularidad para ofrecer nuevas funcionalidades y corregir vulnerabilidades y errores que hayan quedado en su versión anterior. Una de las vulnerabilidades más graves fue la CVE-2015-5143. Esta vulnerabilidad en resumen permitía al atacante remoto no autenticado crear nuevos registros de sesión, esto ocasionaría un llenado del almacenamiento de la sesión provocando la negación de servicio (DoS), provocando la caída del sistema dejando espacio para un posible hackeo en este estado.(NVD, 2015)

Al no recibir actualizaciones el Sistema Integral Informático aparte de no recibir las nuevas funcionalidades está cargado con todas las vulnerabilidades y errores que han sido descubiertos con el paso del tiempo, provocando que su seguridad baje a medida que más vulnerabilidades salen a la luz. Esto sumado a que tener un sistema tan desactualizado entorpece el desarrollo de módulos condicionando así que hoy en día no se desarrollen módulos.

La Dirección Provincial de Informática de Salud Pública en Sancti Spíritus desea volver operativo otra vez al Sistema Integral Informático para integrarlo de nuevo a su organización para el desarrollo de módulos que faciliten el trabajo dentro de sus instalaciones, pero para ello deben resolverse los problemas anteriormente descritos, actualizando su código a los requisitos actuales cambiando su arquitectura de software para hacerlo más flexible.

De la **situación problemática** planteada anteriormente, se deriva el siguiente **Problema de Investigación**.

¿Cómo migrar el software heredado Sistema Integral Informático?

Para solucionar esta situación problemática plantea el siguiente **objetivo general**.

Realizar una migración de software heredado al Sistema Integral Informático.

Partiendo de estos objetivos generales se generan las siguientes **preguntas de investigación**.

1. ¿Qué fundamentos teóricos-metodológicos sustentan la migración de software heredados?
2. ¿Cómo diseñar una migración de software heredado al Sistema Integral Informático?
3. ¿Cómo implementar la migración de software heredado al Sistema Integral Informático?

Las **tareas de investigación** de la presente investigación serían las siguientes.

1. Precisar los fundamentos teóricos-metodológicos sustentan la migración de software heredados.
2. Diseñar una migración de software heredado al Sistema Integral Informático.
3. Implementar la migración de software heredado al Sistema Integral Informático.

La estructura de la presente investigación para dar solución a la problemática se organizó de la siguiente forma: introducción, tres capítulos, conclusiones, recomendaciones, bibliografías y anexos.

En el primer capítulo se abordan los fundamentos teóricos que servirán de base a la presente investigación, se expondrán las tendencias tecnológicas que sustentan el desarrollo de este proyecto. Se analizarán las metodologías que se pueden aplicar a este estudio, así como los framework que se necesitarán para el proceso de migrado del Sistema Integral Informático.

El segundo capítulo se centrará en el diseño de la migración al Sistema Integral Informático, dejando en constancia la factibilidad de realizar la migración y adentrándonos al software para obtener su base de datos y plantear el flujo que tendrá la migración basándose en los estándares de programación que se usarán.

El tercer capítulo se fundamenta en implementar la migración de este sistema basándose en la metodología utilizada, se llevarán a cabo pruebas y una vez superadas se procederá a implementar el sistema migrado.

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS-METODOLÓGICOS QUE SUSTENTAN LA MIGRACIÓN DE SOFTWARES HEREDADOS

Introducción

En este capítulo se darán a conocer las tecnologías, herramientas y metodologías que pudieran ser utilizadas a la hora de realizar esta investigación dejando también conceptos claves para entender su desarrollo.

1.1 Antecedentes

En la actualidad existen varios softwares que permiten el manejo y control de los AFT, a nivel Internacional están los muy conocidos:

Los sistemas de planificación de recursos empresariales o ERP por sus siglas en inglés en la actualidad siguen teniendo gran impacto en el sector empresarial internacional una prueba de ellos es la resistencia al tiempo que poseen dos sistemas que son referentes en lo que a sistemas ERP se refiere, ellos son SAP Business One fundado en el año 2002 y SAGE fundado en 1981 estos y otros sistemas han servido a miles de negocios alrededor del mundo pero algo que casi todos tienen en común es que no pueden hacer contratos con ninguna empresa de nuestro país ya sea por temas de capital o por el Bloqueo impuesto por Estados Unidos a Cuba. Otro punto importante es que las grandes compañías de sistemas ERP manejan un modelo capitalista que no coincide con nuestro modelo lo que hace que a pesar del Bloqueo esta sea otro impedimento.(Gallego, 2020)

Cuba ha hecho frente a esas limitaciones y ha desarrollado estos sistemas empresariales, ejemplos relevantes son el Versat-Sarasola y Rodas XXI que se han esparcido por una gran parte del sistema empresarial. Estos softwares tienen una gran limitante y es que son desarrollados sobre plataformas de software propietario y esto va en contra de la independencia tecnológica que se desea alcanzar en Cuba. Además, son sistemas de escritorio lo cual hace engorroso tener que en cada ordenador que desee acceder al sistema tenga que contar con la aplicación instalada.(Rodríguez, 2016)

1.2 Historia

‘’A inicios de la década de los años 90 se inician proyectos de automatización por el Centro Nacional de Información de Ciencias Médicas (CNICM). Desde las primeras etapas se combinó el desarrollo de redes locales y una red amplia que permitiera la

vinculación del conjunto de instituciones del sistema entre sí y con otras redes nacionales e internacionales'' (Infomed, 2016).

"El proyecto de desarrollar una red para mejorar los servicios de información en el país se da en el contexto de una revolución en esta esfera a nivel internacional, un potencial humano con calificación acumulada durante más de 20 años, la existencia de determinada infraestructura y organización, así como la decisión política de no abandonar los esfuerzos por desarrollar los servicios de información y los esfuerzos para el desarrollo. En este contexto, el 18 de diciembre de 1992 se comenzó a desarrollar en el CNICM una red de computadoras llamada Red Electrónica de Información en Salud (Infomed). Su objetivo entonces era facilitar el intercambio de información entre los profesionales, los académicos, los investigadores y los funcionarios del Sistema Nacional de Salud cubano" (Infomed, 2016).

El surgimiento de esta red vino, por una parte, a dar respuesta a la difícil situación que sufría el país por la aguda crisis económica que lo afectaba desde finales de 1989, la cual conllevó muchas dificultades para adquirir y diseminar la información científico médica y, por otra parte, a posibilitar la asimilación gradual de las modernas tecnologías que se estaban imponiendo respecto a la generación y el uso de nuevos productos y servicios de información. Por ello, desde el mismo momento en que Infomed comenzó a funcionar se orientaron los esfuerzos a ampliar y fortalecer su infraestructura, con una estrategia de progresiva incorporación de servicios soportados en esas tecnologías, que ofrecen alternativas relativamente económicas para diseminar información a muchos grupos de personas con intereses comunes.(Historia, 2022)

El proyecto de desarrollo de la Red Infomed, se ha caracterizado por el uso de las tecnologías de la información y la comunicación con una visión social y a partir del desarrollo de las capacidades locales. Fue la primera red nacional cubana que utilizó el sistema operativo GNU/Linux en todos sus servidores, facilitando el acceso a contenidos nacionales y servicios adecuados a su sistema de salud (Infomed, 2016).

La voluntad del Gobierno cubano y la colaboración recibida, tanto a escala nacional como internacional, han constituido factores clave para la creación y desarrollo de Infomed.

En una nueva etapa de desarrollo de la red, Infomed 2.0 es el nombre que sintetiza la propuesta de asumir una filosofía de trabajo, que se sustente cada vez más en el trabajo colectivo de sus miembros.

“Como parte del proceso de informatización, el departamento informático de Salud Pública Provincial de Sancti Spíritus, perteneciente al Ministerio de Salud Pública (MINSAP), desarrolla varias soluciones informáticas con el fin de automatizar el funcionamiento de las instituciones de salud en la provincia. Entre los proyectos de desarrollo podemos mencionar el Sistema Integral de Información (SII), para la automatización de los procesos informáticos en la provincia”

1.3 Sistema Integral Informático

El Sistema Integral Informático, disponible para todos los informáticos de las unidades de salud de la provincia, es una aplicación web desarrollada con el objetivo de facilitar el trabajo del departamento de informática de Salud Pública provincial de Sancti Spíritus. Cuenta con una serie de módulos que permiten automatizar procesos tales como inspección de departamentos, mensajería, recursos humanos, planificación, entre otros; haciendo más sencillo el trabajo de los trabajadores del departamento previamente referido. Actualmente está fuera de servicio por su grado elevado de desactualización.

1.4 Conceptos claves

1.4.1 Sistema de Gestión de la Información.

Conjunto de herramientas organizativas, técnicas, tecnológicas y de información que se integran en un único sistema para recoger, almacenar, procesar y producir información destinada a realizar funciones de gestión. El sistema de información acumula y procesa la información normativa, de planificación y contable entrante para convertirla en información analítica que sirve de base para prever el desarrollo del sistema de gestión, ajustar los objetivos y planificar un nuevo ciclo de reproducción.(D. C. Pérez, 2022)

1.4.2 Software heredado.

Un software heredado es un sistema, tecnología o aplicación de software antiguo o desactualizado que sigue en uso dentro de una organización porque sigue desempeñando las funciones para las que fue diseñado. Por lo general, los softwares heredados ya no

cuentan con soporte y mantenimiento y están limitados a nivel de crecimiento. Sin embargo, no pueden reemplazarse fácilmente.(Daniel Torres Silva, 2018)

Las empresas pueden detectar si están usando un sistema o aplicación heredada analizando diversos aspectos. Ya que los sistemas pueden quedar obsoletos por varias razones. Por ejemplo, porque el vendedor del sistema discontinúe el producto; lo que se conoce como software heredado EOL (del inglés End-of-life; fin de la vida útil). El producto ya no existe y, por lo tanto, no tiene soporte. Otras razones por las que un sistema puede quedar obsoleto es porque ya no tenga actualizaciones, tenga numerosos parches, no se pueda escalar o no haya personal cualificado dentro de la empresa que sepa cómo funciona.(Daniel Torres Silva, 2018)

Algunos ejemplos de software heredados son SAGE Murano y ContaPlus. Estos 2 sistemas del fabricante SAGE ya han llegado a su fin de vida útil. Lo mismo ocurrirá con SAP Business Suite 7 (R/3 y ECC) en 2030.(Technology, 2021)

Peligros de trabajar con un sistema heredado.

Hay empresas que trabajan con softwares heredados. Esto puede ser porque no tienen la capacidad de inversión (de dinero, tiempo y/o personal) para cambiar de sistema. También puede ser porque crean que actualmente pueden seguir funcionando bien sin mantenimiento, ni soporte y que no van a necesitar añadir licencias. No obstante, al hacerlo deben ser conscientes de una serie de riesgos que asumen al hacerlo:

- **Problemas de seguridad TIC.**
  Los sistemas de software reciben actualizaciones constantemente. Esto sucede por varios motivos. Uno de los principales motivos es porque se descubren errores informáticos en el código (bugs) de los sistemas. Pero, también se realizan actualizaciones porque la tecnología cambia y se requieren nuevas funcionalidades y/o porque pueden darse caídas de sistemas, por ejemplo, de servidores.

  Las actualizaciones y el mantenimiento ayudan a las empresas a mantener su seguridad TIC a prueba de posibles problemas. Para suplir la carencia de estas medidas, hay empresas con software heredados que le ponen “parches” al sistema. Sin embargo, estos parches pueden no ser compatibles con el sistema o dar

problemas de rendimiento. Además, según el sector es posible que los parches deban pasar una auditoría que les dé una valoración positiva para poder usarlos. Por ello, en caso de utilizarlos, es recomendable llevar una gestión de los parches.

- **Falta de compatibilidad con otras herramientas.**
  La tecnología avanza y, consecuentemente, los lenguajes de programación cambian. Esto provoca que un software que fue desarrollado hace 10 años no tenga el mismo lenguaje de programación que uno de reciente creación. Por tanto, si una empresa sigue llevando su sistema ERP con uno heredado y quiere integrarlo con otro software, es muy probable que no pueda.

  Por tanto, el software heredado se convierte en una isla de software. Eso significa que no puede conectarse ni comunicarse con ningún otro sistema, imposibilitando la automatización de procesos.

- **Ausencia de flexibilidad y descenso del rendimiento.**
  Los softwares heredados tienen una serie de desventajas: se vuelven más lentos con el tiempo; no se pueden añadir licencias; ni ampliar módulos; ni funcionalidades. El hecho de que los softwares heredados sean más lentos no es porque se “oxide” el código, sino a que su entorno sí que avanza y se vuelve más exigente. Por tanto, al no actualizarse el sistema, se queda obsoleto en comparación con el entorno y pierde rapidez.(Technology, 2021)

Por otra parte, el no poder añadir funcionalidades ni usuarios impide que la empresa pueda crecer. Con un software heredado se llega a un techo que no se puede sobrepasar. Por lo que, la empresa empieza a incluso perder efectividad debido a no poder automatizar procesos de su actividad por culpa del sistema.(Technology, 2021)

Además, los lenguajes de programación avanzan. Por ello, las empresas con sistemas heredados pierden mucho a nivel de rendimiento. Esto se debe a que la solución, en comparación con una nueva que tiene un lenguaje de programación nuevo, no es tan competitiva ni potente. Por ejemplo, una solución programada con PHP7 será más lenta, menos competitiva y menos potente que una desarrollada con PHP8. Por tanto, la empresa

está en desventaja competitiva frente a otras empresas parecidas que sí que están actualizadas a nivel de software.(Technology, 2021)

1.4.3 Metodología de Software.

Una metodología de desarrollo de software consiste en hacer uso de herramientas, métodos, modelos para el desarrollo y técnicas para estructurar, planear y controlar el proceso de desarrollo de sistemas de información. La gran mayoría de las metodologías tradicionales tales como cascada, prototipos, incremental y espiral suelen ser demasiado documentadas, esto para que los programadores que estarán dentro de la planeación del proyecto puedan entender perfectamente la metodología y en algunos casos el proceso de negocios, razón por la cual hacen pesada cualquier migración.(Daniel Torres Silva, 2018)

Estas metodologías suelen consistir en:

- Realizar un análisis de los requisitos: Consiste en documentar lo que el software deberá realizar al término de su migración.

- Diseño del sistema y programa: Es la realización de un prototipo y los algoritmos a utilizar sin codificar.

- Codificación: Se realiza la escritura del código necesario para el desarrollo del software.

- Ejecución de pruebas: Para valorar y localizar posibles errores o validaciones que no hayan sido consideradas. De igual forma permitirá conocer los tiempos de ejecución y la veracidad de los resultados obtenidos.

- Verificación: Una vez que fue probado por el programador, se instalará para que el usuario realice pruebas reales.

- Mantenimiento del sistema nuevo: Normalmente no se prueban todos los posibles casos por lo que siempre habrá que corregir errores y realizar actualizaciones.

- Amplia documentación en todo momento.(Daniel Torres Silva, 2018)

En cambio, las metodologías agiles como lo es Scrum y Kanban consisten principalmente en:

- Trabajar para obtener software funcional en lugar de demasiada documentación.

- Colaboración con el usuario para la comprensión rápida de sus procesos de negocios.

- Se tiene la posibilidad de hacer cambios de planes en cualquier punto del proyecto evitando la planeación extensa, lo que permite iniciar la programación.(Daniel Torres Silva, 2018)

Sin embargo, aunque sus ventajas son interesantes, estas metodologías también presentan inconvenientes que hay que asumir cuando se decide trabajar con ellas. Estos son:

- Falta de documentación del diseño.

- Problemas de comunicación debido a la participación del usuario donde se pueden dar malas interpretaciones en lo hablado y no documentado.

- Existe una fuerte dependencia de las personas, si el usuario no tiene el tiempo suficiente disponible puede alentar el desarrollo del proyecto.

- Falta de reusabilidad derivada de la falta de documentación.(Daniel Torres Silva, 2018)

1.4.4 Desarrollo Web.

El desarrollo web es un término que define la creación de sitios web para Internet o una intranet. Para lograr esto, la tecnología de software se usa en el lado del servidor y del lado del cliente, lo que implica una combinación de procesos de base de datos utilizando un navegador para realizar ciertas tareas o mostrar información.(Aniel, 2022)

1.4.5 API (Application Programming Interface).

El concepto hace referencia a los procesos, las funciones y los métodos que brinda una determinada biblioteca de programación a modo de capa de abstracción para que sea empleada por otro programa informático. En sí es el código que indica a las aplicaciones

cómo pueden mantener una comunicación entre sí. Estas reglas permiten que los distintos programas mantengan interacciones.(Porto, 2017)

La mayor y más importante función de las API es ayudar a los desarrolladores y hacer que ganen en tiempo y dinero. Por otro lado, también son muy útiles cuando queremos utilizar de manera intencionada las funciones de algún servicio con la intención de que los clientes utilicen nuestra aplicación.(Rojas, 2021)

Ejemplos de las API más utilizadas:

- Google Maps: Nos permite incrustar mapas en nuestro propio sitio utilizando JavaScript pudiendo usar y manipular a nuestra necesidad.

- Youtube: Lo podemos utilizar para crear sitios web y también para apps de escritorio para que consigan información tanto de los usuarios como de los videos.

- Amazon S3: Permite guardar archivos en una infraestructura dada por Amazon con un precio inferior al que nos costaría crear nuestra propia red de servicios.

- Facebook: Uno de los mayores sitios donde más beneficios de negocio podemos encontrar como por ejemplo agencias de viajes gracias a todos los usuarios que tiene. Por otro lado, hace unos años, Facebook se vio obligada a crear una API de código abierto para combatir cierto tipo de propaganda dañina como propaganda terrorista, la explotación infantil o la violencia gráfica.(Rojas, 2021)

Podemos afirmar que hay APIS por todos los lados, aunque nosotros no seamos capaces de identificarlas, otras muchas veces las identificamos, pero no somos conscientes de lo que son.(Rojas, 2021)

1.4.6 Editor de código fuente.

Poderosa herramienta diseñada para desempeñar la tarea específica de escribir y modificar un código web. Los editores de código clásicos son una especie de solución intermedia entre editores de texto simples, y los de entorno de desarrollo integrado IDE por sus siglas en inglés.(Digital, 2022)

Se distinguen de los editores clásicos por una amplia gama de funciones ofrecidas y porque integran múltiples herramientas conectadas para el desarrollo de software. Por lo general, también está integrado en estos entornos de desarrollo complejos.(Digital, 2022)

Las características de los editores de código para varios sistemas operativos son:

- Capacidad de guardar proyectos en cualquier codificación de caracteres
- Coloración de sintaxis para diferentes idiomas (generalmente configurable)
- Función flexible de búsqueda y reemplazo, que también puede usar expresiones regulares
- Función mostrar / ocultar para secciones de código lógicamente relacionadas (plegado de código)
- Finalización automática del código (palabras, funciones, parámetros)
- Editor de macros
- Gestión simplificada de porciones completas de código.(Digital, 2022)

1.4.7 Control de versiones.

El control de versiones, también conocido como "control de código fuente", es la práctica de rastrear y gestionar los cambios en el código de software. Los sistemas de control de versiones son herramientas de software que ayudan a los equipos de software a gestionar los cambios en el código fuente a lo largo del tiempo. A medida que los entornos de desarrollo se aceleran, los sistemas de control de versiones ayudan a los equipos de software a trabajar de forma más rápida e inteligente.(Atlassian, 2022)

Git es un sistema de control de versiones distribuido, lo que significa que un clon local del proyecto es un repositorio de control de versiones completo. Estos repositorios locales plenamente funcionales permiten trabajar sin conexión o de forma remota con facilidad. Los desarrolladores confirman su trabajo localmente y, a continuación, sincronizan su copia del repositorio con la copia en el servidor. Este paradigma es distinto del control de versiones centralizado, donde los clientes deben sincronizar el código con un servidor antes de crear nuevas versiones.(Ignite, 2022)

1.4.8 Modelo Vista Controlador (MVC).

Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, y sobre multitud de lenguajes y plataformas de desarrollo.(Framework, 2022)

El Modelo que contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia. La Vista, o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos interacción con éste. El Controlador, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.(Framework, 2022)

El modelo es el responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.

- Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".

- Lleva un registro de las vistas y controladores del sistema.

- Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero por lotes que actualiza los datos, un temporizador que desencadena una inserción, etc.).(Framework, 2022)

El controlador es responsable de:

- Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).

- Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas

peticiones a las vistas puede ser una llamada al método "Actualizar ()". Una petición al modelo puede ser "Obtener_tiempo_de_entrega (nueva_orden_de_venta)". (Framework, 2022)

Las vistas son responsables de:

- Recibir datos del modelo y los muestra al usuario.

- Tienen un registro de su controlador asociado (normalmente porque además lo instancia).

- Pueden dar el servicio de "Actualización ()", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).(Framework, 2022)

1.4.9 Framework.

Marco o esquema de trabajo generalmente utilizado por programadores para realizar el desarrollo de software. Utilizar un framework permite agilizar los procesos de desarrollo ya que evita tener que escribir código de forma repetitiva, asegura unas buenas prácticas y la consistencia del código. Un framework es por tanto un conjunto de herramientas y módulos que pueden ser reutilizados para varios proyectos.(Arimetrics, 2022)

1.4.10 Arquitectura Rest.

REST no es un protocolo ni un estándar, sino más bien un conjunto de límites de arquitectura. Los desarrolladores de las API pueden implementarlo de distintas maneras. Las API de REST son más rápidas y ligeras, cuentan con mayor capacidad de ajuste.(RedHat, 2020)

características de las API REST y qué las convierte en una herramienta tan útil:

- Protocolo cliente/servidor sin estado:
  Cada petición HTTP contiene toda la información necesaria para ejecutarla, por tanto, esto permite que ni cliente ni servidor necesiten recordar ningún estado previo. No obstante, existen algunas excepciones y hay algunas aplicaciones HTTP

que incorporan memoria caché, para que así, el cliente pueda ejecutar en un futuro la misma respuesta para peticiones idénticas.

- Cuatro operaciones más importantes:
  Las operaciones más importantes relacionadas con los datos en cualquier sistema REST y la especificación HTTP son cuatro; POST (crear), GET (leer y consultar), PUT (editar) y DELETE (borrar).

- Objetos en rest manipulados con uri:
  La URI es el identificador único de cada recurso de un sistema REST. Esta, nos facilita el acceso a la información, para poder modificarla o borrarla. También para compartir su ubicación exacta a terceros.

- Interfaz uniforme:
  Para poder realizar una transferencia de datos en un sistema REST, este aplica acciones concretas (POST, GET, PUT y DELETE) sobre los recursos, siempre y cuando estén identificados con una URI. Esto lo que permite es facilitar la existencia de una interfaz uniforme que sistematiza el proceso con la información.

- Sistema de capas:
  Su estructura o arquitectura es jerárquica entre sus componentes, y cada una de estas capas, lleva a cabo una funcionalidad dentro del sistema REST.

- Utilización de hipermedios:
  El concepto hipermedio utilizado en los casos de API REST sirve para explicar la capacidad de un interfaz de desarrollo de aplicaciones para proporcionar al cliente y al usuario los enlaces adecuados, y ejecutar acciones concretas sobre los datos. Debemos tener en cuenta que cualquier API debe disponer de hipermedios, puesto que este principio es el que define que cada vez que se hace una petición al servidor y este devuelve una respuesta, parte de la información que contendrá serán los hipervínculos de navegación asociada a otros recursos del cliente.(Ribas, 2018)

1.5 Metodología para el desarrollo de software MADIISH

En la actualidad se han diseñado varias metodologías para la migración de un Sistema Heredado (SH), tal es el caso de la metodología de apoyo basada en el uso de las herramientas KDD (Knowledge Discovery in Databases) donde se señala que uno de los factores de éxito de un proyecto de migración es el entendimiento del SH, esto es, entender tanto el modelo de datos como el modelo de negocios que trata de cubrir el mismo. Mediante esta estrategia se pretende reconstruir algunos aspectos básicos del SH a migrar, de modo que sea posible entender el modelo de datos del SH, entender el modelo de negocios que intentaba cubrir el SH y determinar el nivel de calidad de los datos del SH.(Daniel Torres Silva, 2018)

Las propuestas de las metodologías anteriores y la experiencia obtenida durante la migración de un sistema heredado han formado parte de los fundamentos del desarrollo de la metodología MADIISH. La metodología MADIISH está basada en las principales características de las siguientes metodologías:

- Metodología iterativa debido a que se pueden realizar modificaciones constantes sobre un mismo subsistema hasta la entrega satisfactoria al usuario final.

- Metodología incremental ya que se migra subsistema por subsistema.

- Metodología ágil por la participación constante del cliente para reducir la documentación. (Daniel Torres Silva, 2018)

La metodología MADIISH presenta una estructura como se muestra en la siguiente figura:

Ilustración 1 Estructura general de la metodología MADIISH(Daniel Torres Silva, 2018)

Las fases de la metodología MADIISH son las siguientes:

- Fase C: Se define de forma abstracta la conformación del supra sistema en sus diferentes sistemas, así como la de sus sistemas en sub sistemas, lo cual permite establecer la relación entre cada sub sistema para facilitar la determinación de la jerarquía de migración y la interoperabilidad de dicho sub sistema o sistema con otros. Esta fase se denomina fase cero debido a que durante la migración se realiza una vez, al inicio de la migración. A su vez, permite el reconocimiento de las necesidades del cliente y se plantean las estrategias necesarias para poder llevar a cabo el proceso de migración.

- Fase 1: Se elige un sistema a migrar tomando en cuenta los siguientes criterios: o Importancia o Sencillez o Conveniencia o Complejidad o Contingencia o Selectividad A su vez, se realiza un análisis de entradas y salidas de datos para determinar la dependencia de otros sub sistemas. De igual forma se definen los nuevos requerimientos de usuario y se comprende mediante el análisis de procesos de negocio la operación del sistema, esquematizando el conocimiento sobre un diagrama de entradas y salidas de datos, permitiendo la eficiencia de los nuevos procesos.

- Fase 2: Se realiza el modelado la base de datos o actualización de la misma en caso de existir una previa de algún subsistema migrado anteriormente, el desarrollo de interfaces de usuario, la programación para la comunicación entre interfaces con base de datos y la generación de una versión prototipo para realizar pruebas y mostrar al usuario.

- Fase 3: Se implementa un ambiente de pruebas lo más semejante al ambiente de producción en donde se pueda ejecutar un prototipo con información actualizada, si son los resultados esperados se podrá continuar con la siguiente fase, de no serlo se deberá regresar al análisis y esquematización de entradas y salidas de datos. Es aquí otra de las características de MADIISH, permite regresar dentro del ciclo de migración a un punto en el que se pueda determinar las fallas presentadas según el avance obtenido.

- Fase 4: Una vez superadas las fases anteriores es necesario implementar el nuevo sistema en el ambiente de producción por lo que se debe anular el funcionamiento del subsistema heredado migrado, actualizar la base de datos del nuevo sistema e implementar finalmente el nuevo sistema. Se tiene contemplado la habilitación de módulos de interoperabilidad de datos en caso de que la desconexión afecte al flujo de datos entre sistemas.

- Fase 5: Se realiza una documentación del sub sistema o sistema migrado según sea el caso en la cual se describen los procesos de negocios, la documentación del código fuente y la descripción grafica del sistema.

- Fase T: Elaboración de la documentación final, consiste en unir toda la documentación y retroalimentar los manuales o procesos en los que se encuentre ambiguo su contenido.(Daniel Torres Silva, 2018)

1.6 Tecnologías del lado del servidor:

1.6.1 Python 3.10.

Lenguaje de programación de alto nivel que se utiliza para desarrollar aplicaciones de todo tipo. A diferencia de otros lenguajes como Java o .NET, se trata de un lenguaje interpretado, es decir, que no es necesario compilarlo para ejecutar las aplicaciones escritas en Python, sino que se ejecutan directamente por el ordenador utilizando un programa denominado interpretador, por lo que no es necesario “traducirlo” a lenguaje máquina.(Universidades, 2021)

La última versión estable de Python, la 3.10, incluye varias funciones de sintaxis y escritura que los desarrolladores llevaban bastante tiempo esperando que incluyese. Además, los que han trabajado en el desarrollo de esta versión han pulido el intérprete, con el objetivo de que los cambios que han hecho en él hagan que el código de Python tenga una depuración más sencilla.(Valdeolmillos, 2021)

Esta versión tiene también un nuevo operador de unión de tipos, pensado para que los programadores escriban un código más limpio. Además, los mensajes de error que devuelva apuntarán por fin al punto en el que se encuentra el error. Este mensaje también será mucho más claro e informativo que en versiones anteriores, lo que en suma ayudará a los desarrolladores a corregirlo cuanto antes.(Valdeolmillos, 2021)

La principal novedad de esta versión es la estructura de control de flujo por comparación de patrones, que se había caído de versiones anteriores. Con ella, los desarrolladores pueden buscar variables en un conjunto de posibles valores, sin tener que depender de bloques if-else-elif para gestionar los valores de expresión. Varios desarrolladores comparan esta función con el switch/case de otros lenguajes. También permite a los desarrolladores hacer matching en patrones de valores. La función se implementa a través de una declaración de búsqueda que «*toma una expresión y compara su valor a patrones sucesivos dados como uno o más bloques case*» y está diseñada para equipar a los desarrolladores con un sistema sencillo para extraer información de tipos de datos completos.(Valdeolmillos, 2021)

1.6.2 JavaScript.

Lenguaje de programación o de secuencias de comandos que te permite implementar funciones complejas en páginas web, cada vez que una página web hace algo más que

sentarse allí y mostrar información estática para que la veas, muestra oportunas actualizaciones de contenido, mapas interactivos, animación de Gráficos 2D/3D, desplazamiento de máquinas reproductoras de vídeo, etc., puedes apostar que probablemente JavaScript está involucrado. Es la tercera capa del pastel de las tecnologías web estándar.(Mozilla, 2022)

Características de Javascript:

- Lenguaje del lado del cliente:
  Cuando se dice que un lenguaje es del lado del cliente, nos referimos a que se ejecuta en la máquina del propio cliente a través de un navegador. Algunos de estos lenguajes son el propio javascript, HTML, CSS o Java.

- Lenguaje orientado a objetos:
  Javascript es un lenguaje orientado a objetos. Que un lenguaje esté orientado a objetos quiere decir que utiliza clases y objetos como estructuras que permiten organizarse de forma simple y son reutilizables durante todo el desarrollo. Otros lenguajes orientados a objetos son Java, Python o C++.

- De tipado débil o no tipado:
  Que un lenguaje sea de tipado débil quiere decir que no es necesario especificar el tipo de dato al declarar una variable. Esta característica supone una gran ventaja a la hora de ganar rapidez programando, pero puede provocar que cometamos más errores que si tuviéramos esa restricción que poseen los lenguajes de tipado fuerte como C++ o Java.

- De alto nivel:
  Que Javascript sea un lenguaje de alto nivel significa que su sintaxis es fácilmente comprensible por su similitud al lenguaje de las personas. Se le llama de “alto nivel” porque su sintaxis se encuentra alejada del nivel máquina, es decir, del código que procesa una computadora para ejecutar lo que nosotros programamos.

- Lenguaje interpretado:

Javascript es un lenguaje interpretado porque utiliza un intérprete que permite convertir las líneas de código en el lenguaje de la máquina. Esto tiene un gran número de ventajas como la reducción del procesamiento en servidores web al ejecutarse directamente en el navegador del usuario, o que es apto para múltiples plataformas permitiendo usar el mismo código.

- Muy utilizado por desarrolladores:
  A la hora de elegir si aprender o no un nuevo lenguaje, no sólo hay que informarse sobre el tipo de lenguaje o su curva de aprendizaje, sino también su demanda en el mercado. Javascript es en la actualidad uno de los lenguajes más demandados de los últimos años por su versatilidad y su infinita capacidad para crear plataformas cada vez más atractivas.
  Según un estudio de requisitos solicitados en las ofertas de empleo en el año 2020 realizado por la universidad de Boston Northeastern, Javascript es el segundo lenguaje más demandado sólo por detrás de Python.(Miteris, 2022)

De los desarrolladores originales de JavaScript surge la idea de **node.js**. Lo transformaron de algo que solo podía ejecutarse en el navegador en algo que se podría ejecutar en los ordenadores como si de aplicaciones independientes se tratara. Gracias a Node.js se puede ir un paso más allá en la programación con JavaScript no solo creando sitios web interactivos, sino teniendo la capacidad de hacer cosas que otros lenguajes de secuencia de comandos como Python pueden crear.(Lucas, 2019)

Tanto JavaScript como **Node.js** se ejecutan en el motor de tiempo de ejecución JavaScript V8 (V8 es el nombre del motor de JavaScript que alimenta Google Chrome. Es lo que toma nuestro JavaScript y lo ejecuta mientras navega con Chrome). Este motor coge el código JavaScript y lo convierte en un **código de máquina** más rápido. El código de máquina es un código de nivel más bajo que la computadora puede ejecutar sin necesidad de interpretarlo primero, ignorando la compilación y por lo tanto aumentando su velocidad. (Lucas, 2019)

Node.js utiliza un modelo de entrada y salida sin bloqueo controlado por eventos que lo hace ligero y eficiente (con entrada nos referimos a solicitudes y con salida a respuestas). Puede referirse a cualquier operación, desde leer o escribir archivos de cualquier tipo hasta hacer una solicitud HTTP. La finalidad de Node.js no tiene su objetivo en operaciones intensivas del procesador, de hecho, usarlo para programación de más peso eliminará casi

todas sus ventajas. Donde Node.js realmente brilla es en la **creación de aplicaciones de red rápidas,** ya que es capaz de manejar una gran cantidad de conexiones simultáneas con un alto nivel de rendimiento, lo que equivale a una **alta escalabilidad.** (Lucas, 2019)

1.6.3 Django 4.0.

Framework web de alto nivel escrito en Python que fomenta el desarrollo rápido y el diseño limpio y pragmático". Es una herramienta que simplifica la construcción de sitios web desde cero. Al usar licencia BSD, es gratuito y de código abierto.(Guerra, 2014)

Las notas esta versión cubren la abundancia de nuevas funciones en detalle, pero algunos aspectos destacados son:

- El nuevo backend de RedisCache proporciona soporte integrado para el almacenamiento en caché con Redis.

- Para facilitar la personalización de formularios, conjuntos de formularios y listas de errores, ahora se procesan con el motor de plantillas.

- zoneinfo de la biblioteca estándar de Python es ahora la implementación de zona horaria predeterminada en Django.(Felisiak, 2021)

Con el lanzamiento de Django 4.0, Django 3.2 ha llegado al final del soporte principal. Django 3.2 es una versión LTS y recibirá correcciones de seguridad y pérdida de datos hasta abril de 2024. Django 3.1 ha llegado al final del soporte extendido.(Felisiak, 2021)

1.6.4 Django Rest Framework (DRF).

Django Rest Framework es una aplicación Django que permite construir proyectos software bajo la arquitectura REST, incluye gran cantidad de código para reutilizar (Views, Resources, etc.) y una interfaz administrativa desde la cual es posible realizar pruebas sobre las operaciones HTTP como lo son: POST y GET.(Bernal, 2012)

DRF se basa fundamentalmente en 3 componentes: los serializadores, las vistas y los routers.

- Los routers son una herramienta que nos permiten definir las urls de nuestro API de una manera sencilla y ordenada. Básicamente nos permiten definir limpiamente qué método de una class view se ejecutará al llegar una petición HTTP contra un path concreto usando un verbo HTTP u otro. En resumen, nos permiten definir cómodamente conjuntos de urls y nos encaminan a nuestros métodos en función del verbo HTTP (GET, POST, PUT, PATCH...).

- Las views no son más que extensiones de las class-view de django, pero de alguna forma vitaminadas para simplificarnos el enganche con los routers, los serializadores y los modelos y en lugar de renderizar un html como respuesta devolver de forma sencilla un json, xml u otra estructura de datos que nos interese que devuelva nuestra API. En este punto prima la convención sobre la configuración pudiendo basarnos en las clases standards de la librería para describir en muy pocas líneas nuestra API.

- Los serializadores nos permiten definir al detalle cómo serán las respuestas que devolverá nuestro API y cómo procesaremos el contenido de las peticiones que nos lleguen.(Paradigma, 2018)

Algunas de las ventajas que nos ofrece DRF a la hora de implementar APIs son las siguientes:

- API navegable desde el browser lo que agiliza el trabajo de los desarrolladores
- Integración con autenticación basada en OAuth1a o OAuth2.
- Serialización de datos a partir de ORM u otros orígenes.
- Muy buena documentación y una amplia comunidad al ser open source.
- Empresas como RedHat, Heroku o Mozilla lo usan.(Paradigma, 2018)

1.6.5 Vue.js.

Vue es un framework open source de JavaScript, el cual nos permite construir interfaces de usuarios de una forma muy sencilla. La curva de aprendizaje es relativamente baja. Una de las características más importantes de Vue es el trabajo con componentes. Un componente

Vue, en términos simples, es un elemento el cual se encapsula código reutilizable. Dentro de un componente podremos encontrar etiquetas HTML, estilos de CSS y código JavaScript. Los componentes nos permiten desarrollar proyectos modularizados y fáciles de escalar, si nosotros así lo deseamos podemos reemplazar un componente por otro de una forma muy sencilla, como si de piezas de lego se tratasen.(E. I. G. Pérez, 2019)

Las características de Vue son las siguientes:

- La curva de aprendizaje es, con diferencia, la más sencilla de los tres frameworks más populares: React, Vue y Angular.

- Se trata de un framework muy amigable y respetuoso con las tecnologías de frontend y los estándares. Utiliza HTML, CSS y Javascript y es compatible con WebComponents (de hecho, sus componentes se basan en ellos). Si eres desarrollador con base fuerte de HTML/CSS, muy probablemente te guste más Vue que otras opciones.

- Se trata de un framework progresivo. Esto significa que es ideal para migrar y adaptar proyectos existentes hechos en otras tecnologías y pasarlos poco a poco a Vue. Algo muy común cuando trabajas en proyectos heredados (la mayoría de los casos).

- A los desarrolladores que provienen de lenguajes o frameworks exclusivamente de backend están acostumbrados a ciertos patrones de programación que no son exactamente iguales en frontend. Es aconsejable aprender ciertas bases de frontend general. Una buena base de Javascript también es muy recomendable.

- Vue le da mayor protagonismo al enfoque tradicional «centrado en HTML» así como a los sistemas de plantillas. Si te gustan, Vue probablemente te resulte muy atractivo. Por otro lado, el enfoque de React se suele centrar más en programación

pura en Javascript, utilizando HTML y CSS sólo como complementos que se añaden a Javascript.(JS, 2022)

Framework sin opinión.

Vue (al igual que React) se considera una herramienta no opinionated (sin opinión), o lo que es lo mismo, ni Vue ni React te van a guiar por una forma única de hacer las cosas, sino que a lo largo de tus proyectos lo habitual suele ser ir incorporando las características que necesites. También existen múltiples formas de hacer las mismas cosas, cosa que a ciertos desarrolladores les encanta, pero otros lo odian.(JS, 2022)

Por su parte, Angular se considera un framework opinionated (con opinión), ya que es más estricto en cuanto a las decisiones de arquitectura y te obligará en cierta forma a utilizar determinadas tecnologías o características (o al menos te hará más complicado hacerlo de otra forma).(JS, 2022)

Ecosistema de Vue:

El ecosistema de Vue está formado por varias partes, donde cada uno realiza una tarea concreta. Sus partes principales son las siguientes:

Parte	Descripción
Vue	El core o núcleo del framework Vue, donde residen sus funciones principales.
Vue CLI	Asistente para crear y administrar proyectos de Vue desde una terminal o entorno gráfico.
Vue Router	Sistema para crear y gestionar rutas URL desde el navegador en una aplicación de Vue.
Vuex	Gestor de estados para aplicaciones SPA de Vue.

Parte	Descripción
Vue Test Utils	API para realizar tests sobre nuestra aplicación Vue.

Tabla 1 Partes de Vue(JS, 2022)

1.6.6 Nuxt.js

NuxtJS es un framework que se utiliza para el desarrollo de aplicaciones web. Podemos utilizar nuxtJS para crear aplicaciones estáticas (static page), de una sola página (SPA) o de servidor (SSR).(NIGMACODE, 2022)

NuxtJS y Vue.js.

NuxtJS es un framework que trabaja sobre Vue.JS, esto quiere decir que tenemos todo lo bueno de Vue.JS pero contando ya con una organización y configuración establecida desde el principio, que ayuda al desarrollador a enfocarse 100% en el desarrollo desde el principio.(NIGMACODE, 2022)

Por qué utilizar NuxtJS.

- Cuenta con una estructura de carpetas ya definida. Según crece nuestra aplicación, la organización de nuestro código se vuelve de lo más importante y qué mejor que tener ya una estructura definida desde el principio.

- El enrutamiento de NuxtJS es realmente sencillo a comparación con el de Vue.JS. Más adelante del artículo veremos un ejemplo del enrutado de nuxtJs que nos encantará.

- Viene con una configuración lista para producción.

- A diferencia de Vue.JS, NuxtJS cuenta con SSR (server side rendering) para que se indexen las páginas de nuestra web en los buscadores y se optimice el SEO.

- Diferentes modos funcionamiento en nuestra aplicación NuxtJs, podemos hacer paginas estáticas, como ssr como spa.(NIGMACODE, 2022)

1.6.7 Sistema gestor de base de datos(MYSQL)

En programación es prácticamente inevitable trabajar con algún tipo de sistema de gestión de bases de datos. Cualquier programa que imaginemos tarde o temprano necesitará almacenar datos en algún lugar, como mínimo para poder almacenar la lista de usuarios autorizados, sus permisos y propiedades.(Robledano, 2019)

MYSQL es el sistema de gestión de bases de datos relacional más extendido en la actualidad al estar basada en código abierto. Desarrollado originalmente por MYSQL AB, fue adquirida por Sun MicroSystems en 2008 y esta a su vez comprada por Oracle Corporation en 2010, la cual ya era dueña de un motor propio InnoDB para MYSQL.(Robledano, 2019)

MYSQL es un sistema de gestión de bases de datos que cuenta con una doble licencia. Por una parte, es de código abierto, pero por otra, cuenta con una versión comercial gestionada por la compañía Oracle.(Robledano, 2019)

Características de MYSQL:

- Arquitectura Cliente y Servidor: MYSQL basa su funcionamiento en un modelo cliente y servidor. Es decir, clientes y servidores se comunican entre sí de manera diferenciada para un mejor rendimiento.
- Compatibilidad con SQL: SQL es un lenguaje generalizado dentro de la industria. Al ser un estándar MYSQL ofrece plena compatibilidad por lo que si has trabajado en otro motor de bases de datos no tendrás problemas en migrar a MYSQL.
- Vistas: Desde la versión 5.0 de MYSQL se ofrece compatibilidad para poder configurar vistas personalizadas del mismo modo que podemos hacerlo en otras bases de datos SQL. En bases de datos de gran tamaño las vistas se hacen un recurso imprescindible.
- Procedimientos almacenados. MYSQL posee la característica de no procesar las tablas directamente, sino que a través de procedimientos almacenados es posible incrementar la eficacia de nuestra implementación.
- Desencadenantes. MYSQL permite además poder automatizar ciertas tareas dentro de nuestra base de datos.
- Transacciones. Una transacción representa la actuación de diversas operaciones en la base de datos como un dispositivo. (Robledano, 2019)

Ventajas:

Descritas las principales características de MYSQL es fácil ver sus ventajas. MYSQL es una opción razonable para ser usado en ámbito empresarial. Al estar basado en código abierto permite a pequeñas empresas y desarrolladores disponer de una solución fiable y estandarizada para sus aplicaciones. Por ejemplo, si se cuenta con un listado de clientes, una tienda online con un catálogo de productos o incluso una gran selección de contenidos multimedia disponible, MYSQL ayuda a gestionarlo todo debida y ordenadamente.(Robledano, 2019)

1.7 Herramientas para el desarrollo de software

1.7.1 Visual Studio Code.

Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft. Es software libre y multiplataforma, está disponible para Windows, GNU/Linux y macOS. VS Code tiene una buena integración con Git, cuenta con soporte para depuración de código, y dispone de un sinnúmero de extensiones, que básicamente te da la posibilidad de escribir y ejecutar código en cualquier lenguaje de programación.(Flores, 2022)

Ventajas de Visual Studio Code:

- Visual Studio Code es una herramienta que tiene soporte nativo para gran variedad de lenguajes, entre ellos podemos destacar los principales del desarrollo Web: HTML, CSS, y JavaScript, entre otros.

- Adaptar el visual al gusto del usuario. De esta forma, podremos tener más de un código visible al mismo tiempo, las carpetas de nuestro proyecto y también acceso a la terminal o un detalle de problemas, entre otras posibilidades. (Luca, 2022)

1.7.2 GitHub

GitHub es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador. La plataforma está creada para que los desarrolladores suban el código de sus aplicaciones y herramientas, y que como usuario no sólo puedas descargarte la aplicación, sino también entrar a su perfil para leer sobre ella o colaborar con su desarrollo. Como su nombre indica, la web utiliza el sistema de control de versiones Git.(Fernández, 2019)

Las principales características de la plataforma es que ofrece las mejores características de este tipo de servicios sin perder la simplicidad, y es una de las más utilizadas del mundo por los desarrolladores. Es multiplataforma, y tiene multitud de interfaces de usuario. Además de permitirte mirar el código y descargarte las diferentes versiones de una aplicación, la plataforma también hace las veces de red social conectando desarrolladores con usuarios para que estos puedan colaborar mejorando la aplicación.(Fernández, 2019)

Conclusiones parciales.

Durante la investigación de la presente investigación y lo abordado en este capítulo se puede llegar a las siguientes conclusiones parciales:

- Se definió la importancia de migrar los softwares heredados a las tecnologías actuales.

- Se escogió como metodología más adecuada a la presente investigación la MADIISH por ofrecer ventajas sobre la migración de software heredados.

- Se establecieron todos los recursos necesarios para llevar a cabo la migración y el desarrollo de las API y el frontend, seleccionando para las API la aplicación de Django, Django Rest Framework y para el frontend el framework Nuxt.js.

- Se escogió como editor de código fuente a Visual Studio Code por sus ventajas expuestas.

- Se usará el portal de control de versiones GitHub

CAPÍTULO 2: DISEÑO DE LA MIGRACIÓN DE SOFTWARE HEREDADO DEL SISTEMA INTEGRAL INFORMÁTICO.

Introducción

Con el marco teórico ya definido y escogidas las herramientas y metodología a utilizar, se está en condiciones de comenzar el desarrollo del siguiente capítulo. En este capítulo se desarrollan las fases de planificación y diseño propias de la metodología propuesta para el desarrollo del sistema.

2.1 Factibilidad técnica

La Dirección Provincial de Informática de Salud Pública en Sancti Spíritus cuenta con los recursos necesarios para realizar la migración del Sistema Heredado:

- Servidores con procesamiento suficiente para sostener al nuevo sistema.
- Computadoras modernas que soportan las tecnologías que se van a manejar:
  - Navegadores actualizados
  - Velocidades de conexión superiores a los 100 M/bits

2.2 Factibilidad operacional

Se cuenta con recursos humanos capacitados para manejar al nuevo sistema, no obstante, se creará un pequeño espacio para impartir un curso de capacitación para el personal que no esté capacitado

2.3 Factibilidad económica-financiera

Con la incorporación del nuevo sistema a la empresa se agilizarán las gestiones económicas y de recursos humanos ya que el sistema cuenta con módulos para el manejo logístico, económico, personal, etc.

2.4 Especificación de requerimientos

Propósito

Con la migración de este sistema se traerá devuelta un sistema integral que facilitará muchas de las funciones dentro de la empresa, además de tener la opción de incorporar nuevos módulos para nuevos requerimientos que lo necesiten, se llevarán con mayor control las gestiones de la entidad, así como tener un espacio para guardar imágenes, videos de actividades o de cualquier momento que decida la institución.

2.4.1 Ámbito del software.

MIGRACIÓN DEL SOFTWARE HEREDADO SISTEMA INTEGRAL INFORMÁTICO DE LA DIRECCIÓN PROVINCIAL DE INFORMÁTICA DE SALUD PÚBLICA EN SANCTI SPÍRITUS

Esta migración beneficiará directamente al departamento de informática ya que se separará la lógica del sistema y la de la interfaz mediante el uso de API, esto implicará mejor rendimiento, facilidad de mantenimiento y seguridad del sistema. En la parte de los usuarios que usarán el sistema les ahorrará mucho tiempo y complejidades a la hora de realizar sus tareas, se ahorrarán materias primas como lo es el papel y se ganará en organización. Como nuevas prestaciones están:

- Nueva interfaz basada en el uso de API
- Optimización de funciones obsoletas pertenecientes a versiones anteriores de django
- Optimización de la estructura de los modelos con los nuevos cambios de django 4.0
- Desarrollo de módulo enlaces para el departamento de informática

2.4.2 Funciones del producto.

Justificación de la migración

Entre los principales motivos por los cuales se plantea hacer esta migración de Sistemas Heredados están los siguientes:

- Inexistencia de tecnologías actuales que puedan encajar en el software heredado.
- Imposibilidad de realizar una actualización ya que desde los módulos bases está heredado el sistema.
- Altos costos y tiempo de llevar un nuevo proyecto que sustituya al actual.

Procedimiento propuesto

La presente migración llevará a cabo la siguiente serie de pasos para ser realizada:

- Determinación de la causa de la migración.
- Recolección de datos claves del elemento a migrar
- Consulta a los usuarios y explicación de las razones por las que se va a llevar a cabo la migración y cómo les afectará.
- Planificar el momento y procedimiento de migración.
- Evaluar los resultados de la migración.

Estrategia de migración

Pilares para la migración del sistema:

- Metodología.
- Herramientas
- Métodos de pruebas y personalización.

La metodología garantiza que haya un orden y un plan a seguir, haciendo posible la división de responsabilidades, control sobre los avances y reunir todo lo necesario para llevar a cabo la migración, tanto en personal como en tecnología.

Las herramientas son claves en el plan de llevar a cabo la migración, ellas ofrecen apoyo indispensable para minar datos, acortar tiempo en análisis como a la hora de buscar las relaciones entre los elementos dentro del modelo del sistema, entre otras muchas facultades.

2.4.3 Características de los usuarios del sistema

Nombre de usuario	Tipo de Usuario	Área funcional	Actividad
Administrador	Administrador del Sistema	Administración	- Administrar el sistema. - Administrar cuentas. - Añadir atributos. - Añadir características. - Control integro de los datos.
Usuario de Informática	Usuario del sistema	Informática	- Administrar departamentos - Administrar las plazas de los departamentos - Control de recursos humanos - Administrar enlaces

Tabla 2 Características de los usuarios del sistema

2.4.4 Restricciones del desarrollo

- Como gestor de Base de Datos se usará MYSQL
- El Sistema no permitirá crear usuarios fuera del Django admin
- Se usará exclusivamente el servicio de API para mostrar información de la interfaz

- El lenguaje a usar para la interfaz será escrito en javascript usando el Framework Nuxt.js + vuetify.js
- Los recursos de iconos serán de mdi-icons
- El Sistema solo será de uso interno dentro de la organización

2.4.5 Requisitos

Funcionales

- Toda la información que se manejará en el modelo del sistema será a través de API
- La interfaz mostrará la información que se envié a través de la API
- El resto de los requisitos se mantienen como el original

No funcionales

- Eficiencia:
  - El sistema trabajará en 2 servidores distintos, uno de la lógica y otro de la interfaz aumentando la velocidad de respuesta mediante la comunicación de API**.**
  - El Sistema al contar con tecnología actualizada aumentará mucho su velocidad de respuesta.
- Usabilidad:
  - El sistema contará con la nueva interfaz, manteniendo las características visuales acordes al original.
  - El Departamento de Informática dispondrá de un curso para los trabajadores que no estén asociados al trabajo en este sistema.
- Seguridad:
  - Solo personal asociado a la institución tendrá acceso al sistema y al manejo de la información y todas sus acciones quedarán registradas en el sistema.
  - Solo el administrador e informático podrán eliminar información sensible en el sistema.
- Mantenibilidad:
  - El sistema estará montado en la versión más reciente de django actualmente, la 4.0 tendrá soporte hasta 2026 su versión LTS, se actualizará el sistema hasta llegar a la versión LTS y a partir de ahí a futuro se planteará una nueva migración de así requerirlo.

- Se le podrán agregar módulos y funciones que se adapten al contexto en que se plantee.

2.5 Esquema de la base de datos

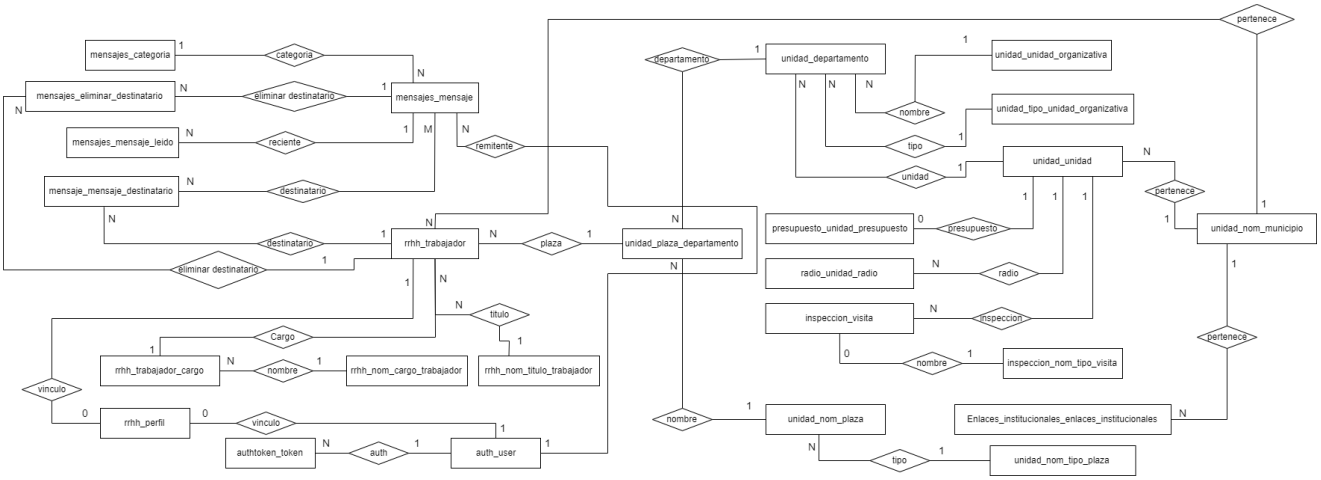


Ilustración 2 Esquema de la base de datos

2.6 Diseño de la Interfaz

Ilustración 3 Diseño de la interfaz

2.7 Estándares de programación utilizados

Creación de las API

Por defecto las API de cada módulo deberán almacenarse en una carpeta dentro del módulo llamada api, dentro deberán existir su vista correspondiente al igual que su serializador y urls.

Los serializadores deberán comenzar con el nombre del modelo correspondiente seguido y finalizando con ListSerializer y/o DetailSerializer en caso de ser serializador de un objeto específico. Todo separado por un guion bajo.

Cada vista debe comenzar con el nombre del modelo correspondiente y finalizando con el nombre del método que fue usado para su creación. El identificador específico debe ser el id del objeto a buscar.

Cada url de cada módulo debe comenzar con el nombre del modelo al que hace referencia, seguido del módulo al que pertenece, todo separado por un guion bajo.

En el archivo urls principal del sistema deben añadirse la referencia de todas las urls de las API almacenadas en los módulos y deben comenzar con “/api/”.

Creación de páginas en el frontend

El título del encabezado de la página tiene que comenzar con el nombre de la página que hace referencia seguido de “| Sistema Integral Informático.

Se recomienda usar la plantilla base para comenzar a crear una página que está alojada en la raíz del backend, se titula “plantilla base de apps del sii a usar en vue.txt”. En ella están pre hechos ya los requisitos básicos de seguridad con que debe contar la página.

Los nombres de las paginas deben comenzar en minúscula.

Nombrar métodos

Los métodos deben comenzar con minúsculas y solo se permite usar el carácter especial guion bajo para separar si es una palabra compuesta.

Nombrar variables

Todas las variables, incluidas los v-models de vue deben comenzar con minúsculas y en caso de ser una palabra compuesta separarla mediante un guion bajo.

Usar solo los v-models para elementos que se mostrarán en pantalla o serán reutilizados por varios métodos, los demás casos usar variables javascript dentro del método correspondiente.

Los arreglos, listas y objetos que contengan más de un valor deben ser nombrados en plural.

Comentarios

Los comentarios deben aplicarse de manera moderada en puntos claves para facilitarle la interpretación al desarrollador y orientarlo para facilitar su trabajo.

Conclusiones parciales

En este capítulo se pudo establecer la metodología más adecuada para la migración del sistema de la presente investigación, se establecieron las causas de por qué era necesario realizar una migración y por qué usar la metodología MADIISH, se plantearon los requisitos, condiciones y estrategia para la migración además de que se logró sacar todos los usuarios del sistema y sus características, se expuso las restricciones del sistema y la factibilidad de realizar la migración, se plantearon las nuevas funcionalidades y los cambios que tendrá el sistema migrado, además de establecer la base de datos general del sistema y los estándares de programación que se usarán.

CAPÍTULO 3: IMPLEMENTACIÓN DE LA MIGRACIÓN DE SOFTWARE HEREDADO DEL SISTEMA INTEGRAL INFORMÁTICO.

Introducción

Con la metodología establecida, las causas de la migración y los requisitos preliminares para comenzar la migración, se da paso a la implementación de la migración del Sistema Integral Informático donde se comenzarán a desarrollar las fases de la metodología MADIISH

3.1 Implementación

Se tomará MADIISH como metodología para la migración de softwares heredados para este proyecto porque abarca buena parte de todos los procesos de trabajo a desarrollar y cuenta con un plan guía enfocado en etapas contempladas con anterioridad.

Durante una de las etapas de desarrollo se utilizará la metodología XP como método de desarrollo ágil.

Se deben realizar acciones necesarias para lograr la migración del sistema en cuestión y luego desarrollar y documentar los pasos requeridos para lograr esta migración, se deben establecer planes de acción para los usuarios, administradores y sistema a migrar.

3.1.1 Datos generales del proyecto

Nombre de la institución:	Dirección Provincial de Informática de Salud Pública de Sancti Spíritus.
Nombre del proyecto:	Sistema Integral Informático

Tabla 3 Datos generales del proyecto

3.1.2 Datos del sistema/software a migrar

Nombre versión del sistema/software a migrar:	Django 1.6: Sistema Integral Informático (versión 1)
Alternativa a implementar	Django 4.0 + Nuxt.js: Python, JavaScript

Tabla 4 Datos del sistema/software a migrar

Con la metodología ya definida para el proceso de migración del Sistema Integral Informático se procede a iniciar las fases de la metodología MADIISH.

3.1.3 Fase C

Se define de forma abstracta la conformación del supra sistema en sus diferentes sub sistemas para realizar la migración iterativa a cada uno de estos sistemas, lo que permite establecer una relación entre cada sub sistema para facilitar la determinación de jerarquía de migración y la interoperabilidad de dicho sub sistemas con otros. Esta fase se denomina fase cero debido a que solo se realizará una vez durante toda la migración.

Como supra sistema quedo definido el Sistema Integral Informático.

Como sub sistemas fueron definidos los siguientes:

- Sub sistema Base: Este subsistema conforma la base del Sistema Integral Informático. Es el encargado de darle acceso al usuario a los módulos del sistema a los que esté autorizado a acceder, contiene la información del usuario activo y todas las gestiones que puede hacer dicho usuario además que diversos módulos están directamente vinculados a este sub sistema.
- Sub sistema autenticación: Este sub sistema se encarga de la seguridad de acceso a la información del sistema, valida que el usuario que esté intentando acceder al sistema realmente esté asociado al sistema
- Sub sistema Mensajes: Este sub sistema contiene estrecha relación con el sub sistema base pues una parte lo integra directamente el sistema Base. Se encarga de mantener la comunicación entre los usuarios del sistema.
- Sub sistema Departamento: Es el encargado de gestionar los departamentos que posee una unidad asociada al sistema. No posee relación directa con el sub sistema base
- Sub sistema Plazas: Se encarga de gestionar las plazas que posee un departamento asociado al sistema. Tiene estrecha relación con el sub sistema Departamento
- Sub sistema RRHH: Es el encargado de gestionar los trabajadores que integran un departamento de una unidad asociada al sistema. Tiene estrecha relación con el sub sistema Plazas
- Sub sistema Enlaces: Se encarga de gestionar los enlaces institucionales de la provincia de Sancti Spíritus. No posee relación directa con ningún sub sistema

Prioridades de migración:

- Sub sistema Base: Alta
- Sub sistema autenticación: Alta

- Sub sistema Mensajes: Alta
- Sub sistema Enlaces: Alta
- Sub sistema Departamento: Media
- Sub sistema Plazas: Media
- Sub sistema RRHH: Media

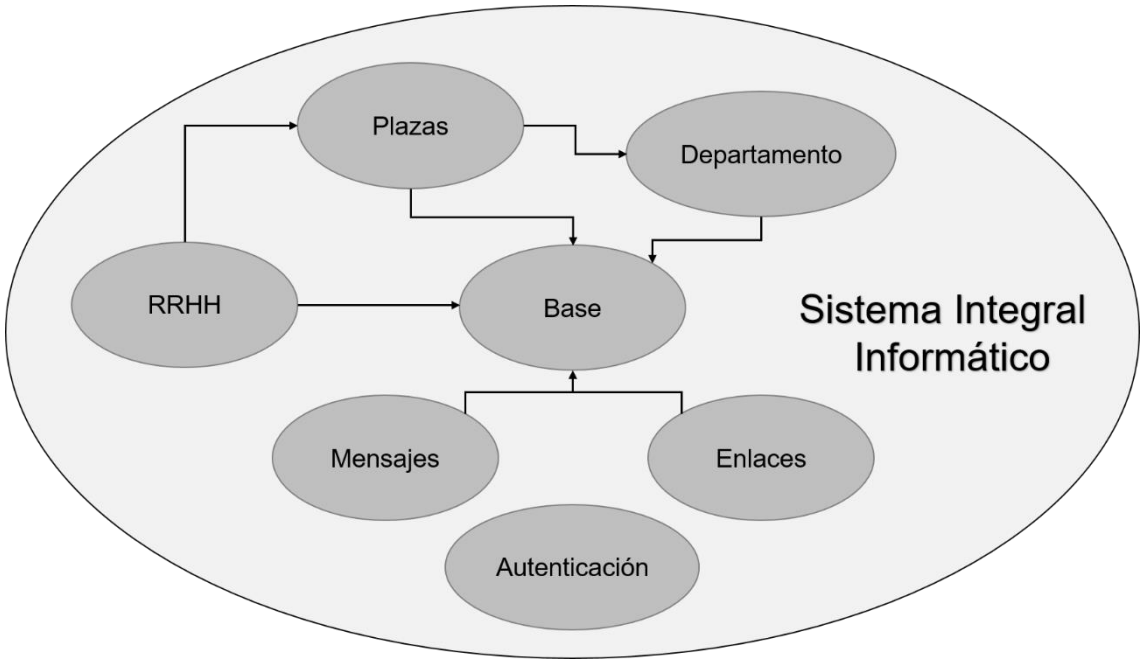


Ilustración 4 Gráfico del diseño de los sub sistemas del Sistema Integral Informático

3.2 Sub sistema Base

3.2.1 Fase 1

Como primer sub sistema a migrar fue seleccionado el Base, este subsistema es de la más alta prioridad ya que es la interfaz principal del usuario y donde hace sus gestiones y todos los sub sistemas dependen de él. Se realizará un análisis de entradas y salidas para ver cuál es la dependencia con los demás subsistemas. Se definirá los nuevos requerimientos del usuario para este sub sistema. Se procede entonces a realizar el esquema de entradas y salidas de datos en un diagrama.

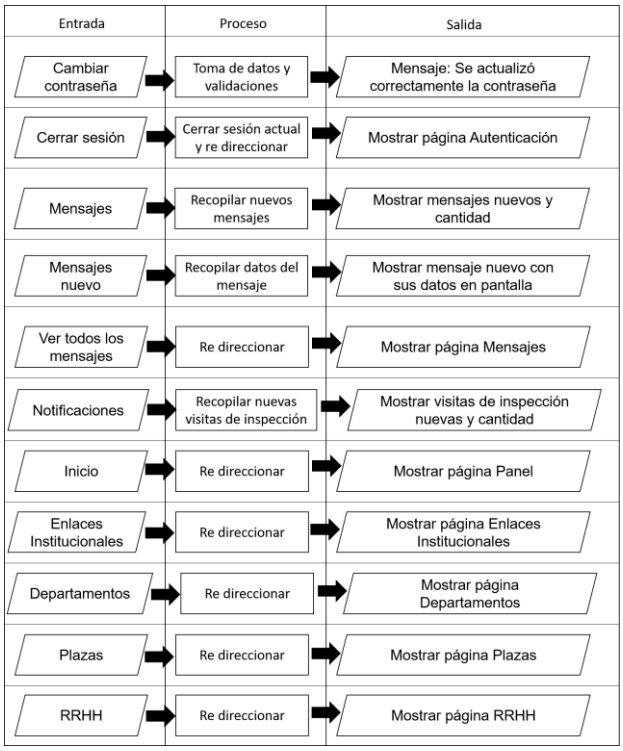


Ilustración 5 Diagrama de entradas y salidas del sub sistema Base

Las dependencias de este sub sistema quedaron reflejadas en el diagrama de entradas y salidas de datos. Estas son:

- Sub sistema autenticación
- Sub sistema Mensajes
- Sub sistema Enlaces Institucionales
- Sub sistema Departamentos
- Sub sistema Plazas
- Sub sistema RRHH

Nuevos requerimientos del usuario

- Foto de perfil de usuario redondeada
- Tema oscuro
- Menú lateral expandible

3.2.2 Fase 2

Modelado de la base de datos:

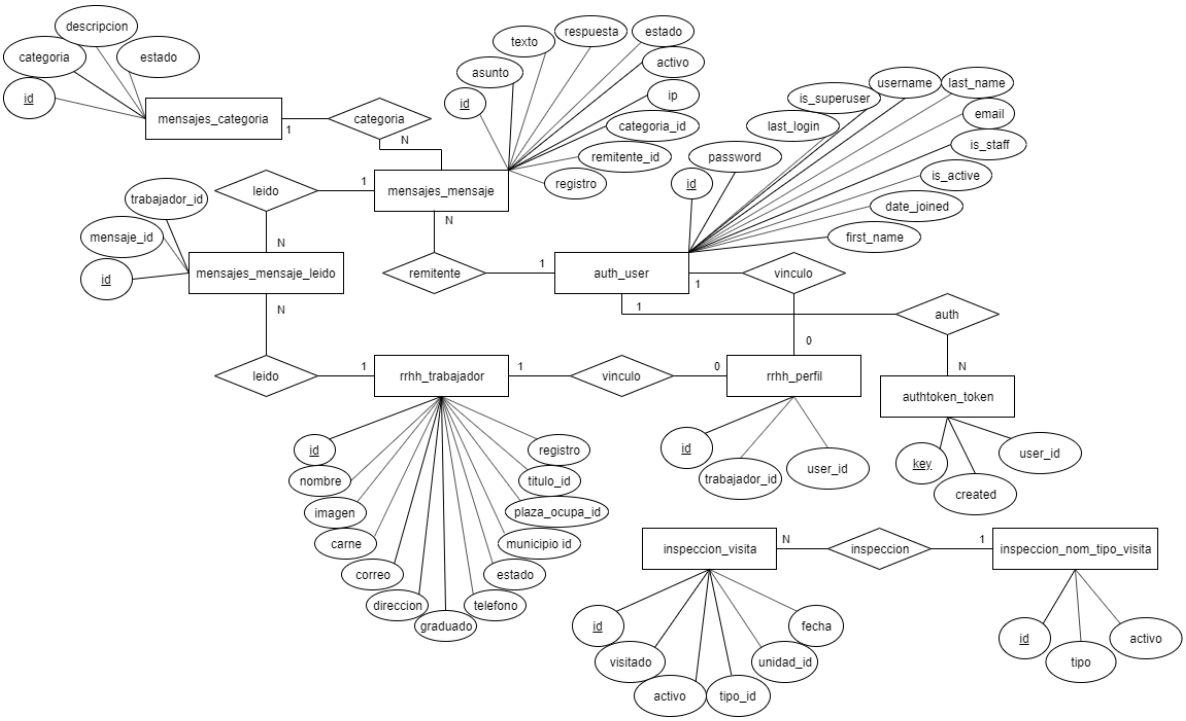


Ilustración 6 Esquema de la base de datos para el sub sistema Base

La base de datos para este sub sistema mantiene sus características originales.

Interfaces de usuario

Panel. Antiguo/nuevo

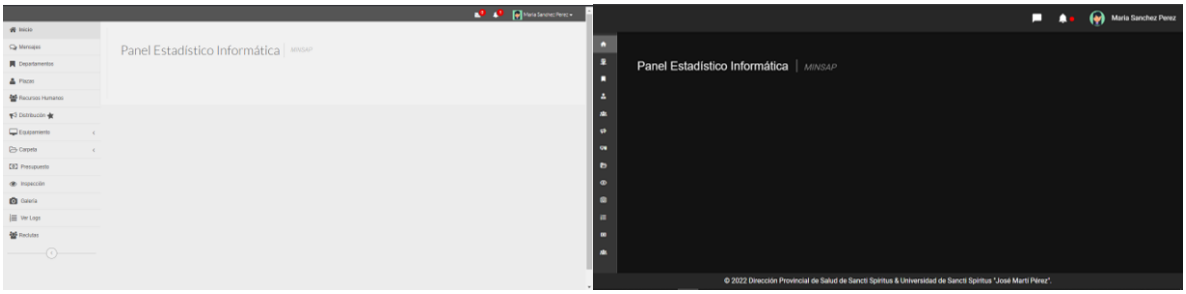


Ilustración 7 Interfaz Panel sub sistema Base

Perfil desplegado. Antiguo/nuevo

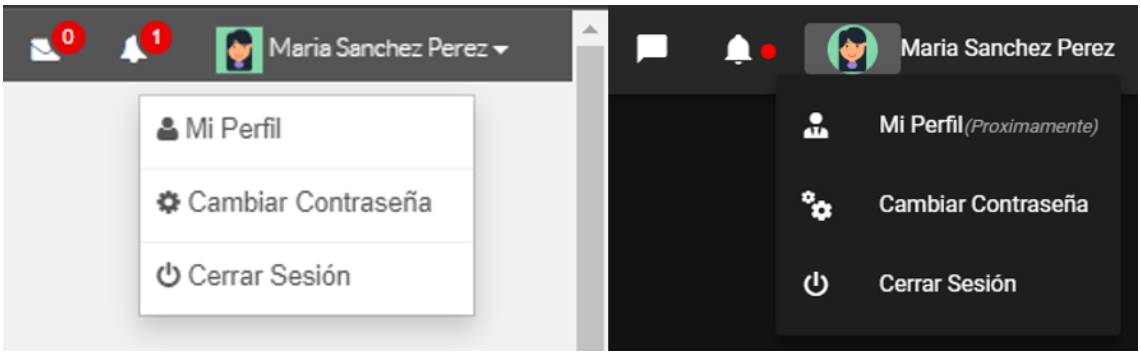


Ilustración 8 Interfaz Perfil sub sistema Base

Cambiar contraseña. Antiguo/actual

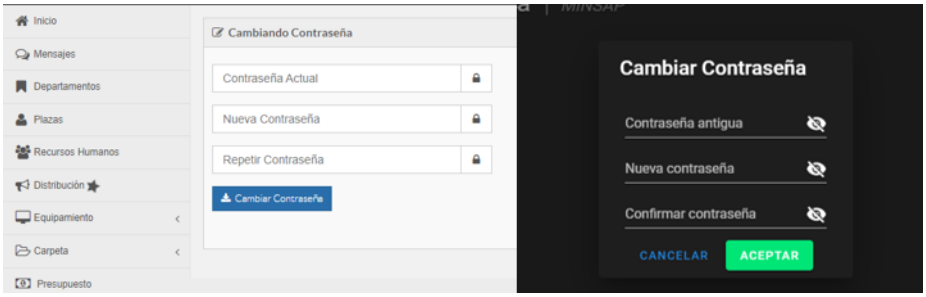


Ilustración 9 Interfaz Cambiar contraseña sub sistema Base

Notificaciones. Antiguo/nuevo

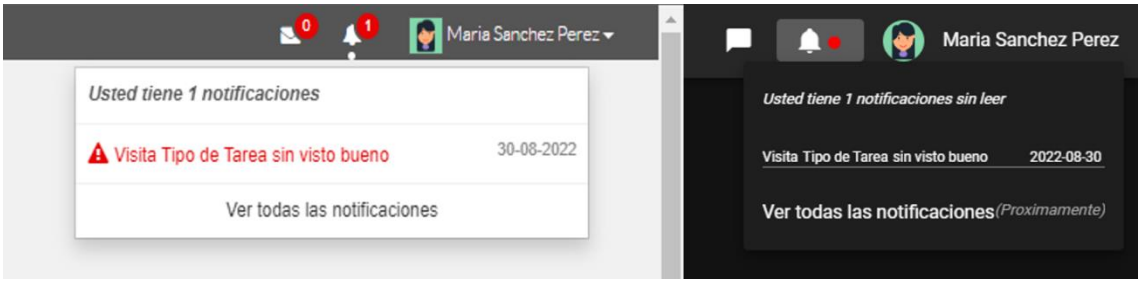


Ilustración 10 Interfaz Notificaciones sub sistema Base

Mensajes. Antiguo/nuevo

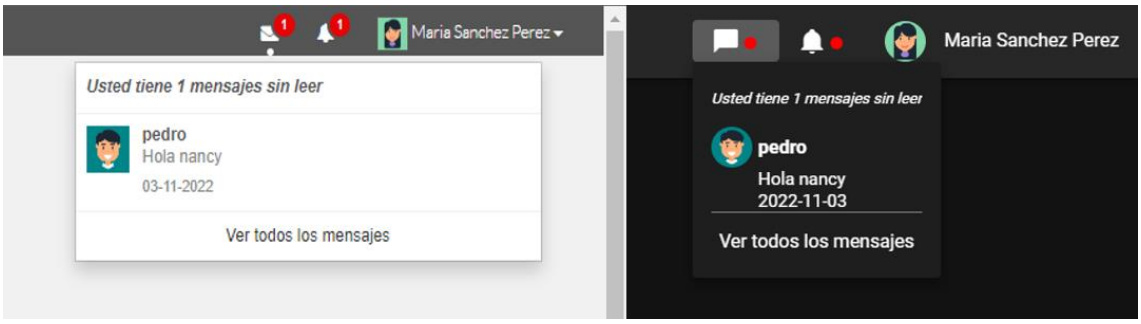


Ilustración 11 Interfaz Mensajes sub sistema Base

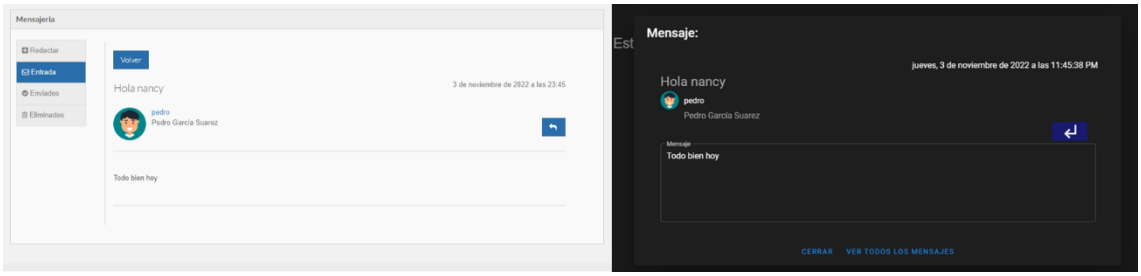


Ilustración 12 Interfaz Mensaje reciente sub sistema Base

Comunicación con la interfaz

Panel estadístico Frontend

```
data() {
  return {
    unidad_organizativa:'',
    unidad:'',
  }
},

methods:{
  async initialize() {
    if(sessionStorage.getItem("usuario")==null){this.$router.push({path:'/', replace:true})}
    this.unidad_organizativa=sessionStorage.getItem("unidad_organizativa")
    this.unidad=sessionStorage.getItem("unidad")
  }
}
```

Ilustración 13 frontend Panel sub sistema Base

Perfil Backend/Frontend

```python
class TrabajadorGAPIViewList(generics.GenericAPIView, mixins.ListModelMixin, mixins.CreateModelMixin, mixins.RetrieveModelMixin):
    serializer_class= serializers.TrabajadorListSerializer

    queryset= models.Trabajador.objects.all()

    lookup_field= 'id'
    def get(self, request, id=None):
        if id:
            return self.retrieve(request)
        else:
         return self.list(request)

    def post(self, request):
        return self.create(request)


class TrabajadorGAPIViewDetail(generics.GenericAPIView, mixins.ListModelMixin, mixins.UpdateModelMixin, mixins.RetrieveModelMixin, mixins.DestroyModelMixin):
    serializer_class= serializers.TrabajadorListSerializer

    queryset= models.Trabajador.objects.all()

    lookup_field= 'id'
    def get(self, request, id=None):
        if id:
            return self.retrieve(request)
        else:
         return self.list(request)


    def put(self, request, id=None):
        return self.update(request,id)

    def delete(self, request, id):
        return self.destroy(request,id)
```

Ilustración 14 backend Perfil sub sistema Base

```javascript
await axios.get(this.$axios.defaults.baseURL+"Trabajador_rrhh/" + sessionStorage.getItem("trabajador") + "/",
{ headers: { "Authorization": "token " + sessionStorage.getItem("token") } })
    .then((result) => {
    this.trabajador = result.data;
}) //nombre del trabajador actual para que salga al lado de la foto de perfil
    .catch((response) => { console.log(response); });
```

```
<div v-if="trabajador" class="content">
<span v-if="trabajador.imagen==null">
  <v-icon >mdi-account-tie</v-icon>
</span>
<span v-else>
  <img :src=trabajador.imagen alt="" style="height:35px; width:35px; object-fit:cover; border-radius: 50%;">
</span>
</div>
```

```
<div v-if="trabajador" class="content">
{{trabajador.nombre}}
  </div>
```

Ilustración 15 frontend Perfil sub sistema Base

Cambiar contraseña Backend/Frontend

```python
#Cambiar contraseña
class ChangePasswordView(generics.UpdateAPIView):
    """
    An endpoint for changing password.
    """
    serializer_class = serializers.ChangePasswordSerializer
    model = User

    def get_object(self, queryset=None):
        obj = self.request.user
        return obj

    def update(self, request, *args, **kwargs):
        self.object = self.get_object()
        serializer = self.get_serializer(data=request.data)

        if serializer.is_valid():
            # Check old password
            if not self.object.check_password(serializer.data.get("old_password")):
                return Response({"old_password": ["Wrong password."]}, status=status.HTTP_400_BAD_REQUEST)
            # set_password also hashes the password that the user will get
            self.object.set_password(serializer.data.get("new_password"))
            self.object.save()
            response = {
                'status': 'success',
                'code': status.HTTP_200_OK,
                'message': 'Se actualizó correctamente la contraseña',
                'data': []
            }

            return Response(response)

        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

Ilustración 16 backend Cambiar contraseña sub sistema Base

```javascript
cambiar_password() {
    this.loading = true;
    axios.put(this.$axios.defaults.baseURL+"change_password/", {
        old_password: this.old_pass,
        new_password: this.new_pass
    }, { headers: { "Authorization": "token " + sessionStorage.getItem("token") } })
        .then(response => {
        alert(response.data.message);
        this.cancel_pass();
    })
        .catch((response) => {
        if (response == "Error: Request failed with status code 400") {
            alert("Contraseña antigua incorrecta, por favor verifiquela y vuelva a intentarlo");
            this.old_pass = "",
                this.new_pass = "",
                this.confirm_pass = "",
                this.loading = false;
        }
        else {
            alert(response);
        }
        console.log(response);
    });
},
```

Ilustración 17 frontend Cambiar contraseña sub sistema Base

Cerrar Sesión Backend/Frontend

```python
#Cerrar sesión
class Logout(APIView):
    def get(self, request, format=None):
        request.user.auth_token.delete()
        return Response(status=status.HTTP_200_OK)
```

Ilustración 18 backend Cerrar sesión sub sistema Base

```javascript
if (item.title == "Cerrar Sesión") {
    clearInterval(this.interval);

    axios.get(this.$axios.defaults.baseURL+"logout/", { headers: { "Authorization": "token " + sessionStorage.getItem("token") } });
    sessionStorage.clear();
    this.$router.push({ path: "/", replace: true });
}
```

Ilustración 19 frontend Cerrar sesión sub sistema Base

Notificaciones Backend/ Frontend

```python
# para el manejo de los tipos de visita
class Tipo_VisitaGAPIViewList(generics.GenericAPIView, mixins.ListModelMixin, mixins.CreateModelMixin, mixins.RetrieveModelMixin):···


class Tipo_VisitaGAPIViewDetail(generics.GenericAPIView, mixins.ListModelMixin, mixins.UpdateModelMixin, mixins.RetrieveModelMixin, mixins.DestroyModelMixin):···
```

Ilustración 20 backend Notificaciones sub sistema Base

```javascript
    this.tipo_visita = [];
  await axios.get(this.$axios.defaults.baseURL+"Tipo_Visita_inspeccion/", { headers: { "Authorization": "token " + sessionStorage.getItem("token") } })
      .then((response) => {
      response.data.forEach((item) => {
          this.tipo_visita.push({
              id: item.id,
              tipo: item.tipo,
          });
      });
  });
```

Ilustración 21 frontend Notificaciones sub sistema Base

Mensajes Backend/Frontend

```python
#elementos para el manejo de las categorias de los mensajes
@api_view(['GET','POST'])

def Categoria_Mensaje_List(request):···



@api_view(['GET','PUT','Delete'])

def Categoria_Mensaje_detail(request, pk):···


#elementor para el manejo de los mensajes
@api_view(['GET','POST'])

def Mensaje_List(request):···


@api_view(['GET','PUT','Delete'])

def Mensaje_detail(request, pk):···


#elementor para el manejo de los mensajes leidos
@api_view(['GET','POST'])

def Mensaje_Leido_List(request):···


@api_view(['GET','PUT','Delete'])

def Mensaje_Leido_detail(request, pk):···
```

```javascript
await axios.get(this.$axios.defaults.baseURL+"Categoria_mensajes/", { headers: { "Authorization": "token " + sessionStorage.getItem("token") } })
    .then((response) => {
    response.data.forEach((item) => {
        categorias_sms.push({
            id: item.id,
            categoria: item.categoria
        });
    });
});
```

```javascript
this.mensajes = [];
await axios.get(this.$axios.defaults.baseURL+"Mensaje_mensajes/", { headers: { "Authorization": "token " + sessionStorage.getItem("token") } })
    .then((response) => {
    response.data.forEach((item) => {
        if (item.destinatario.includes(parseInt(sessionStorage.getItem("trabajador"))) && item.estado == 0) { // Se almacenan en un array temporal
            mensajes_temp.push({
                id: item.id,
                fecha: item.registro,
                categoria: item.categoria,
                asunto: item.asunto,
                enviado: item.remitente,
                estado: item.estado,
                texto: item.texto,
                respuesta: item.respuesta,
                activo: item.activo,
                ip: item.ip,
                destinatario: item.destinatario
            });
        }
    });
});
```

```javascript
async mensajes_leidos(item) {
    var id = item.id;
    await axios.post(this.$axios.defaults.baseURL+"Mensaje_Leido_mensajes/", {
        mensaje: id,
        trabajador: sessionStorage.getItem("trabajador")
    }, { headers: { "Authorization": "token " + sessionStorage.getItem("token") } });
    //////////Datos del mensaje reciente que preciono ahora el usuario para que se muestre en el dialogo/////
    this.editedIndex = this.mensajes.indexOf(item);
    this.mensajes.splice(this.editedIndex, 1);
    this.dialogSMS = true;
    this.id = item.id;
    this.imagen = item.imagen;
    this.asunto = item.asunto;
    this.username_r = item.enviado;
    this.trabajador_r = item.trabajador;
    this.texto = item.texto;
    this.remitenteID = item.remitente;
    var options = { weekday: "long", year: "numeric", month: "long", day: "numeric" };
    var fecha = new Date(item.fecha_full).toLocaleDateString("es-ES", options);
    var hora = new Date(item.fecha_full).toLocaleTimeString();
    this.fecha_dialog = fecha;
    this.hora = hora;
    var url = window.location.href;
    if (url.indexOf("mensajes/contacto") != -1) {
        this.des_reenvio = true;
    }
    else {
        this.des_reenvio = false;
    }
},
```

Ilustración 22 backend Mensajes sub sistema Base/frontend Mensajes sub sistema Base

Menú lateral Frontend

```
mounted(){
  ////////////////////Menu lateral////////////////////
  if (sessionStorage.getItem("unidad_organizativa") == "Informática") {
    this.items.push(
      { icon: "mdi-ip-network", title: "Enlaces Institucionales", to: "/enlaces" },
      { icon: "mdi-bookmark", title: "Departamentos", to: "/departamentos" },
      { icon: "mdi-account-tie", title: "Plazas", to: "/plazas" },
      { icon: "mdi-account-group", title: "Recursos Humanos", to: "/rrhh" },
      { icon: "mdi-bullhorn-outline", title: "Distribución", to: "" },
      { icon: "mdi-desktop-tower-monitor", title: "Equipamiento", to: "" },
      { icon: "mdi-folder-open-outline", title: "Carpeta", to: "" },
      { icon: "mdi-eye-outline", title: "Inspección", to: "" },
      { icon: "mdi-camera-outline", title: "Galería", to: "" },
      { icon: "mdi-format-list-numbered", title: "Ver Logs", to: "" },
    );
```

Ilustración 23 frontend Menú lateral sub sistema Base

Comportamiento del sub sistema con BDD

```
Feature: Cerrar Sesión

    Scenario: Cierre de sesión del usuario actual
    Given Navegador con el panel del Sistema integral Informatico asumiendo que este autenticado con el perfil desplegado
    When El usuario toque cerrar sesión
    Then El sistema cierra la sesión del usuario
    And El sistema redirecciona al usuario a la pantalla de autenticación

Feature: Cambiar contraseña
    Scenario:Cambio de contraseña del usuario actual
    Given Navegador con el panel del Sistema integral Informatico asumiendo que este autenticado con el perfil desplegado
    When El usuario toque cambiar contraseña, rellene los datos solicitados y  ooprima aceptar
    Then El sistema verifica los datos y los procesa
    And Muestra un mensaje de cambio de contraseña exitoso
    But Los datos no fueron validos, mostrar mensaje aclaratorio

Feature: Mensajes

    Scenario: Sección de mensajes en el panel
    Given Navegador con el panel del Sistema integral Informatico asumiendo que este autenticado
    When El usuario toque el botón de mensajes
    Then Mostrar mensajes recientes

    Scenario: Mensajes nuevos deplegados
    Given Navegador con el panel del Sistema integral Informatico asumiendo que este autenticado y con los mensajes recientes desplegados
    When El usuario toque un mensaje reciente
    Then Mostrar la información de ese mensaje
    And Eliminar de la lista de mensajes recientes
```

Ilustración 24 Comportamiento del sub sistema Base con BDD 1

```
Feature: Notificaciones

    Scenario: Sección de notificaciones en el panel
    Given Navegador con el panel del Sistema integral Informatico asumiendo que este autenticado
    When el usuario toque el botón notificaciones
    Then Mostrar notificaciones recientes

Feature: Menu lateral
    Scenario: Menú lateral del sistema integral informatico
    Given Navegador con el panel del Sistema integral Informatico asumiendo que este autenticado
    When El puntero pasa sobre el lateral izquierdo
    Then Desplegar el menú lateral

    Scenario: Menú lateral del sistema integral informatico desplegado
    Given Navegador con el panel del Sistema integral Informatico asumiendo que este autenticado
    When el usuario toque alguna función del menú lateral
    Then re direccionarlo hacia el destino de esa función
```

Ilustración 25 Comportamiento del sub sistema Base con BDD 2

3.2.3 Fase 3

Pruebas

Se realizarán pruebas para poner a prueba la fiabilidad del prototipo del sub sistema en cuestión. Serán pruebas funcionales y no funcionales. Se incluirán pruebas de rendimiento, carga, estrés, mantenibilidad, fiabilidad o portabilidad.

Finalizando las pruebas se debe dejar constancia de que se cumplió las expectativas y se pudo contribuir a la solución del problema.

Pruebas de funcionalidad (aceptación del usuario)

Tomado Cambiar contraseña como muestra para el Test plan pruebas

Realizado Por: Ing. Julio Companioni Martínez

Ambiente de revisión: TEST

Detalles de la aplicación:

Nombre: Sistema Integral Informático

Plataforma: Django 4.0 + Nuxt.js

Base de datos: MYSQL

Desarrollado Por: Franco Salgado

Opciones de revisión

Perfil

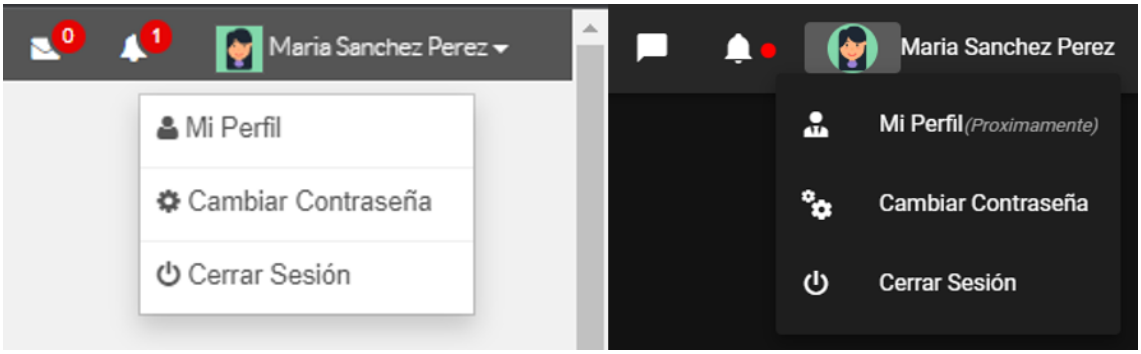


Ilustración 26 Interfaz Perfil sub sistema Base (TEST)

Opción Cambiar contraseña

Insertar	**No tiene**
Actualizar	Correcto
Eliminar	No tiene

Tabla 5 Opción Cambiar contraseña (TEST)

Visualización

El cuadro de dialogo es demasiado ancho, reducirle su anchura

Agregar un botón a cada campo para ver la contraseña

Aumentar tamaño del encabezado

Validación

No se detectaron errores

Funcionabilidad

Cuando el usuario cancele y no quiera cambiar su contraseña, vaciar dialogo para que cuando abra nuevamente no aparezca sus datos anteriores

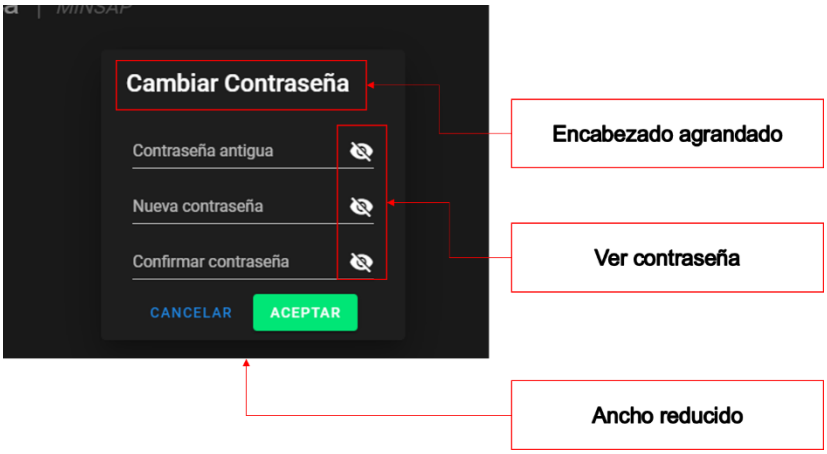


Ilustración 27 Interfaz Cambiar contraseña sub sistema Base (TEST)

Tomado Mensajes recientes como muestra para el Test plan pruebas

Realizado Por: Ing. Julio Companioni Martínez

Ambiente de revisión: TEST

Detalles de la aplicación

Nombre: Sistema Integral Informático

Plataforma: Django 4.0 + Nuxt.js

Base de datos: MYSQL

Desarrollado Por: Franco Salgado

Opciones de revisión

Mensajes

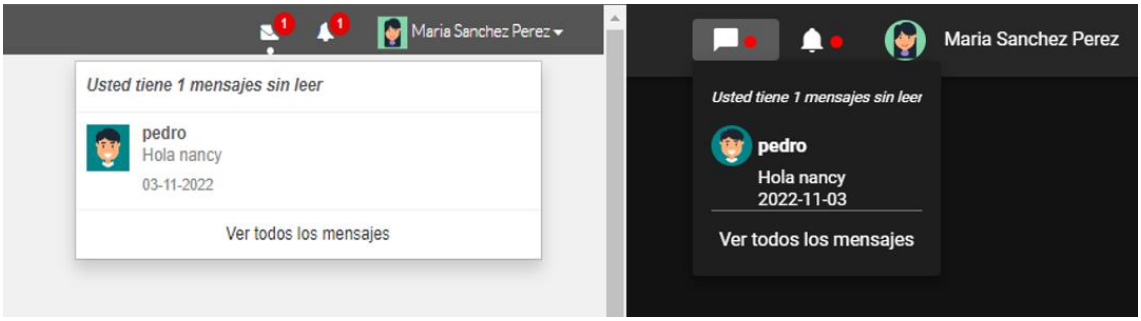


Ilustración 28 Interfaz Mensajes sub sistema Base (TEST)

Opción Mensaje reciente

Insertar	**Correcto**
Actualizar	No tiene
Eliminar	No tiene

Tabla 6 Opción Mensaje reciente(TEST)

Visualización

Poner texto Usted tiene x mensajes en cursiva y reducir su tamaño.

Aumentar un poco el tamaño de la foto del remitente.

Funcionalidad

Recortar el asunto para que no desborde el diálogo.

La foto separarla del nombre y los otros campos realizando una tabla de 2 columnas.

Al momento de abrir el mensaje automáticamente sacarlo de la lista de mensajes nuevos sin actualizar.

Corregir la función responder, para que pase a través de una dirección los datos del mensaje.

Pruebas de rendimiento (aceptación técnica)

Ver mensaje reciente: se procede a ver el mensaje reciente

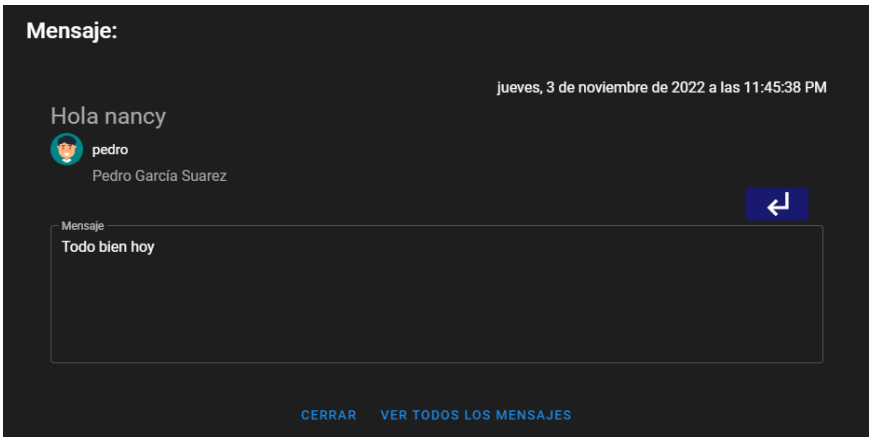


Ilustración 29 Interfaz Mensaje reciente sub sistema Base (TEST)

Tiempo de respuesta en 169 milésimas de segundos para mostrar el mensaje y 421 para postear que ese mensaje fue leído ya.

Ilustración 30 tiempo de respuesta mensaje reciente (TEST)

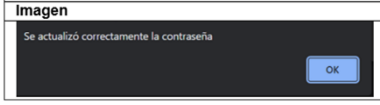
FORMATO PLAN DE PRUEBAS DE USUARIO			
Identificador	PSB01	**Versión**	V01
Responsable	Ing. Julio Companioni Martínez		
Nombre del caso de prueba	Cambiar contraseña		
Módulo		**Sub módulo**	
Base			
Formulario			
/Panel			
Descripción de la prueba			
Ingresar contraseña antigua Ingresar nueva contraseña Ingresar repetir contraseña Aceptar			
Resultados esperados			
Contraseña actualizada			
Resultados reales			
Se actualizó correctamente la contraseña			
Error			
Imagen			
			

Tabla 7 Prueba PSB01

FORMATO PLAN DE PRUEBAS DE USUARIO			
Identificador	PSB02	**Versión**	V01
Responsable	Ing. Julio Companioni Martínez		
Nombre del caso de prueba	Mensajes recientes		
Módulo		**Sub módulo**	
Base			
Formulario			
/Panel			
Descripción de la prueba			
Acceder al icono de mensaje Acceder a un mensaje reciente Cerrar mensaje reciente			
Resultados esperados			
Ver el mensaje y eliminar de la lista de mensajes dicho mensaje			
Resultados reales			
Se mostró el mensaje y se eliminó el mensaje de la lista de recientes			
Error			
Imagen			
			

Tabla 8 Prueba PSB02

3.2.4 Fase 4

Se precede a implementar el sub sistema terminado en un ambiente de producción. Como actualmente el Sistema Integral Informático no está prestando servicio no es una complicación ponerlo en producción. En las fases anteriores quedó definido su modelo, su esquema BDD y se procede ahora a realizar la migración de la base de datos y el sub sistema.

3.2.5 Fase 5

Se procede a dejar en constancia los requisitos de implementación del presente sub sistema, estos son:

Software para el servidor

- Node.js v16.14.2
- Python 3.10

Módulos de Django

- Django Rest Framework
- Django-formtools
- Django-admin-interface
- Django-cors-headers
- ReportLab

Software para usuarios

- Navegador web (Chrome, Firefox)

Hardware para el servidor (recomendado)

- RAM al menos 8GB
- Procesador I5
- Tarjeta Red 1Gb/s

Hardware para el usuario

- RAM 2GB

Manual de usuario

Anexos

3.3 Sub sistema Autenticación

El desarrollo de este sub sistema se encuentra en el Anexo 1.

3.4 Sub sistema Mensajes

El desarrollo de este sub sistema se encuentra en el Anexo 2.

3.5 Sub sistema Enlaces

El desarrollo de este sub sistema se encuentra en el Anexo 3.

3.6 Sub sistema Departamento

El desarrollo de este sub sistema se encuentra en el Anexo 4.

3.7 Sub sistema Plazas

El desarrollo de este sub sistema se encuentra en el Anexo 5.

3.8 Sub sistema RRHH

El desarrollo de este sub sistema se encuentra en el Anexo 6.

Conclusiones parciales

En el desarrollo de este capítulo fueron confeccionados los diversos sub sistemas que componían al Sistema Integral Informático, para dar paso a la implementación de las fases que componen a la metodología MADIISH, se llevó a cabo el desarrollo de cada uno de

estos sub sistemas finalizando con una serie de pruebas para validar su correcto desempeño para luego implementarlos en la institución.

CONCLUSIONES

Con la realización del presente proyecto se arribaron las siguientes conclusiones:

1- La realización del marco teórico permitió ampliar el conocimiento acerca de los softwares heredados, adentrarnos en sus características y ver el por qué es necesario en este caso que el Sistema Integral Informático no siga siéndolo. También se reconoció la metodología MADIISH como metodología recomendada para la migración de este tipo de softwares.

2- En el diseño de la presente migración se dejó constancia de la factibilidad en diferentes aspectos que tiene esta migración para llevarse a cabo, se especificaron los requerimientos, el esquema general de la base de datos del sistema a migrar, así como una muestra del diseño que tendrá la interfaz del sistema y por último los estándares de programación que fueron utilizados en la implementación.

3- Se llevó a cabo el proceso de implementación del Sistema Integral Informático migrado, este nuevo sistema responde al uso de API, con una interfaz renovada basada en Nuxt.js, un framework javascript que cuenta con soporte constante, la versión escogida para Django fue la 4.0 que todavía queda años de soporte y una futura versión LTS. Este sistema quedó con los módulos principales del departamento de informática confeccionados, estos son, recursos humanos, plazas y departamentos. A petición del cliente fue confeccionado un nuevo módulo que responde a la situación actual del departamento para manejar los enlaces institucionales. Con la implementación del sistema se abren las puertas para el desarrollo de nuevos módulos que respondan a las necesidades actuales del Sistema de Salud Pública.

RECOMENDACIONES

Como recomendaciones se estimula a:

- Implementar la sección de notificaciones, orientándola al uso que se requiera en la institución.
- En el módulo de enlaces poner una máscara a los campos de ip para mejorar la experiencia del usuario.
- Implementar un switch para cambiar de modo oscuro a claro para ajustar el sistema a las preferencias del usuario.
- Implementar información de perfil en el menú desplegable del perfil del usuario actual.

BIBLIOGRAFÍA

Aguilar, L. J. (2015). *Sistemas de Información en la Empresa* Alfaomega (Ed.) Retrieved from https://altametrics.com/es/information-systems.html

Aniel. (2022). ¿Qué es la programación web y para que sirve? , from https://www.aniel.es/desarrollo-web/programacion-web/#:~:text=Definici%C3%B3n%20de%20desarrollo%20web&text=Para%20lograr%20esto%2C%20la%20tecnolog%C3%ADa,ciertas%20tareas%20o%20mostrar%20informaci%C3%B3n.

Arimetrics. (2022). Framework. from https://www.arimetrics.com/glosario-digital/framework

Atlassian. (2022). ¿Qué es el control de versiones? , from https://www.atlassian.com/es/git/tutorials/what-is-version-control

Bernal, J. P. R. (2012). Introducción a Django Rest Framework. from https://axiacore.com/blog/introduccion-a-django-rest-framework-460/

Cadre, H. C. (2017). *SISTEMAS DE INFORMACIÓN EMPRESARIAL. EVOLUCIÓN HISTÓRICA Y ACTUALIDAD.* Universidad de Ciego de ÁvilaMáximo Gómez Báez. Retrieved from https://www.gestiopolis.com/sistemas-informacion-implicacion-cuba/

Daniel Torres Silva, J. D. O. G., Héctor Andrade Gómez, Rafael Rivera López. (2018). Una Propuesta de Metodología para la

Migración de Sistemas Heredados *Instituto Tecnológico de Veracruz*, 10.

Digital, P. (2022). Editores de código: ¿Qué son y para qué sirven?

Felisiak, M. (2021). Django 4.0 released. from https://www.djangoproject.com/weblog/2021/dec/07/django-40-released/

Fernández, Y. (2019). Qué es Github y qué es lo que le ofrece a los desarrolladores. from https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores

Flores, F. (2022). Qué es Visual Studio Code y qué ventajas ofrece. from https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/

Framework, S. d. I. N. M. (2022). Modelo vista controlador (MVC). from https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html

Gallego, E. I. (2020). ¿Cuáles son los ERP más extendidos en la actualidad? , from https://www.elidealgallego.com/texto-diario/mostrar/2342648/cuales-erp-extendidos-actualidad

Guerra, J. R. (2014). Descubre qué es Django, el framework web de moda. from https://computerhoy.com/noticias/internet/descubre-que-es-django-framework-web-moda-8641

Historia. (2022). from http://www.sld.cu/acerca-de?quicktabs_pagina_acercade=1#quicktabspagina_acercade

Ignite, M. (2022). ¿Qué es Git? , from https://learn.microsoft.com/es-es/devops/develop/git/what-is-git

Infomed. (2016). Historia. from http://www.sld.cu/acerca-de?quicktabs_pagina_acercade=1#quicktabspagina_acercade

JS, L. (2022). ¿Qué es Vue? , from https://lenguajejs.com/vuejs/introduccion/que-es-vue/

Luca, D. d. (2022). Visual Studio Code: características principales. from https://damiandeluca.com.ar/visual-studio-code-caracteristicas-principales

Lucas, J. (2019). Qué es NodeJS y para qué sirve. from https://openwebinars.net/blog/que-es-nodejs/

Lugones, E. H. (2016). Los sistemas de información y su implicación para Cuba. from https://www.gestiopolis.com/sistemas-informacion-implicacion-cuba/

Miteris. (2022). ¿Qué es Javascript? Características y Librerías. from https://www.miteris.com/blog/que-es-javascript-caracteristicas-librerias/

Mozilla. (2022). ¿Qué es JavaScript? , from https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript

NIGMACODE. (2022). Qué es NuxtJS y Por qué utilizarlo. from https://www.nigmacode.com/javascript/que-es-nuxtjs/

NVD. (2015). CVE-2015-5143 Detail. from https://nvd.nist.gov/vuln/detail/CVE-2015-5143

Paradigma. (2018). Introducción a Django REST framework., from https://www.paradigmadigital.com/dev/introduccion-django-rest-framework/

Pérez, D. C. (2022). ¿Qué es un sistema de gestión de la información? , from https://www.ceupe.com/blog/que-es-un-sistema-de-gestion-de-informacion.html?dt=1655676929369

Pérez, E. I. G. (2019). ¿Qué es Vue.JS? , from https://codigofacilito.com/articulos/que-es-vue

Porto, J. P. (2017). DEFINICIÓN DE

API. from https://definicion.de/api/

RedHat. (2020). ¿Qué es una API de REST? , from https://www.redhat.com/es/topics/api/what-is-a-rest-api#:~:text=Una%20API%20de%20REST%2C%20o,de%20estado%20representacional%20(REST).

Ribas, E. (2018). Qué es Api Rest y por qué debes de integrarla en tu negocio. from https://www.iebschool.com/blog/que-es-api-rest-integrar-negocio-business-tech/

Robledano, A. (2019). Qué es MYSQL: Características y ventajas. from https://openwebinars.net/blog/que-es-MYSQL/

Rodríguez, L. P. (2016). *Aplicación Web para la gestión de los Activos Fijos Tangibles y gestión de los Servicios Técnicos y Transporte en el Departamento de Informática Provincial de Salud Pública en Sancti Spíritus.*, José Martí Pérez.

Rojas, Á. (2021). ¿Que es una API? ¿Para qué sirven las API? , from https://www.incentro.com/es-ES/blog/que-es-api-interfaz-programacion-aplicaciones

Technology, E. K. C. f. I. (2021). ¿hasta cuándo se puede seguir trabajando con un sistema heredado? , from https://www.ticportal.es/glosario-tic/sistema-legacy

Universidades, S. (2021). Python: qué es y por qué deberías aprender a utilizarlo. from https://www.becas-santander.com/es/blog/python-que-es.html

Valdeolmillos, C. (2021). Llega Python 3.10 con varias novedades y mejoras muy esperadas. from https://www.muycomputerpro.com/2021/10/07/python-3-10-novedades

Viana, E. D. P. (2015). *BIG DATA, UNA HERRAMIENTA FUNDAMENTAL PARA CONOCER AL CLIENTE.* SANTO TOMÁS. Retrieved from https://www.dynamicgc.es/historia-del-big-data/#:~:text=y%20aplicaci%C3%B3n%20actual.-,Nacimiento%20del%20t%C3%A9rmino,los%20t%C3%A9rminos%20que%20actualmente%20conocemos

ANEXOS

Anexo 1: Sub sistema Autenticación

Fase 1

Entrada	Proceso	Salida
Aceptar	Toma de datos, validar acceso	Mostrar Página Panel
Ver contraseña	Cambiar de tipo password a String	Mostrar u ocultar contraseña

Ilustración 31 Diagrama de entradas y salidas del sub sistema Autenticación

Este sub sistema no posee dependencias:

Fase 2

Modelado de la base de datos:

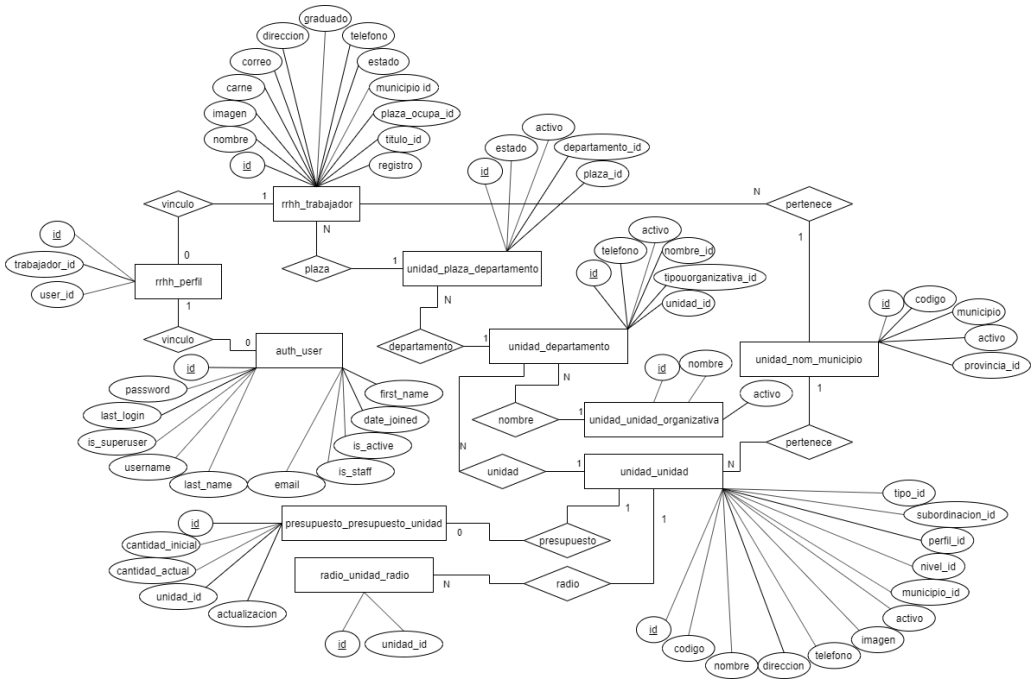


Ilustración 32 Esquema de la base de datos para el sub sistema Autenticación

Interfaces de usuario

Autenticación. Antiguo/nuevo

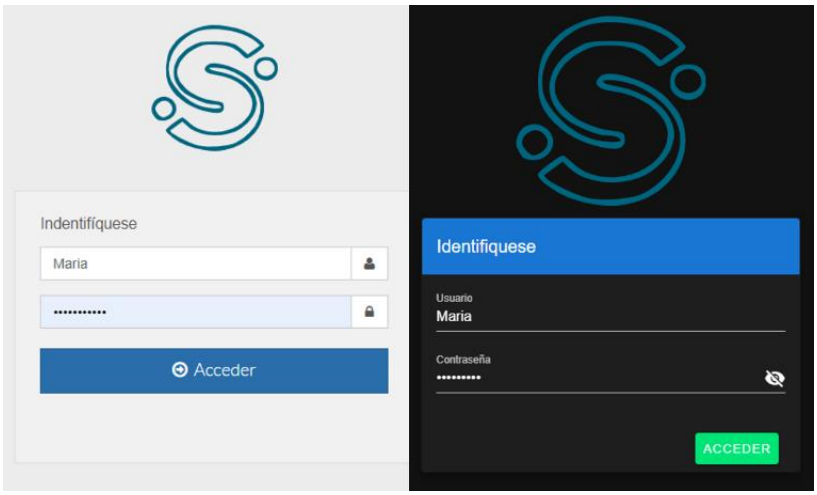


Ilustración 33 Interfaz autenticación sub sistema Autenticación

Comunicación con la interfaz

Backend/Frontend

```
login(){
  this.color='primary'
  this.loading=true
  var bodyFormData = new FormData();
  bodyFormData.append('username', this.user);
  bodyFormData.append('password', this.password);

  axios({
    method: "post",
    url: this.$axios.defaults.baseURL+"login/",
    data: bodyFormData,
    headers: { "Content-Type": "multipart/form-data" },})
    .then((response) =>{
      this.token=response.data
      if(this.token.token==null && this.token!='contraseña incorrecta' && this.token!='usuario desconocido' ){setTimeout(()=>this.login(),1000) }})
    .catch((response) =>{
      if(response=='Error: Network Error'){ this.error='Servidor fuera de servicio'}
      else{this.error=response}
      this.color='red'
      this.loading=false
      console.log(response)});
  this.loading=true
  setTimeout(()=>this.login_next(),2000)
},

async login_next(){
```

```
#autenticacion
@api_view(['POST'])
@permission_classes([~IsAuthenticated])
def login(request):
    username=request.POST.get('username')
    password=request.POST.get('password')


    try:
        user=User.objects.get(username=username)
    except User.DoesNotExist:
        return Response('usuario desconocido')

    pwd_valid=check_password(password,user.password)

    if not pwd_valid:
        return Response('contraseña incorrecta')

    token, _=Token.objects.get_or_create(user=user)
   # token =Token.objects.create(user=user)
    is_expired, token = token_expire_handler(token)



    return Response({
        'token': token.key,
        'expires_in':expires_in(token),})
```

Ilustración 34 backend/frontend autenticación sub sistema Autenticación

Comportamiento del sub sistema con BDD

```
# sub sistema Auth
Feature: Autenticación
    Scenario: Autenticación del sistema integral informático
    Given Navegador con el Sistema integral Informatico en la pantalla de autenticación
    When Cuando el usario llene sus datos y precione acceder
    Then Validar información
    And Re direccionar al panel del Sistema Integral Informático
```

Ilustración 35 Comportamiento del sub sistema Autenticación con BDD

Fase 3

Pruebas de funcionalidad (aceptación del usuario)

Tomado Autenticarse como muestra para el Test plan pruebas

Realizado Por: Ing. Julio Companioni Martínez

Ambiente de revisión: TEST

Detalles de la aplicación:

Nombre: Sistema Integral Informático

Plataforma: Django 4.0 + Nuxt.js

Base de datos: MYSQL

Desarrollado Por: Franco Salgado

Opciones de revisión

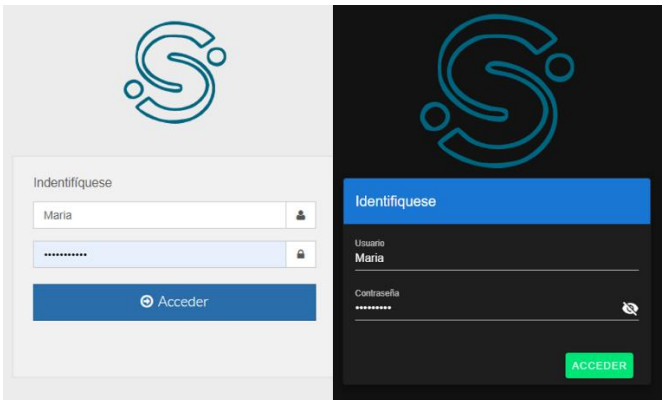


Ilustración 36 Interfaz autenticación sub sistema Autenticación (TEST)

Opción Autenticarse

Insertar	**Correcto**
Actualizar	No tiene
Eliminar	No tiene

Tabla 9 Opción Autenticarse (TEST)

Visualización

Botón acceder mostrar en la esquina inferior derecha.

Centrar logo del Sistema Integral Informático.

Validación

Cundo el servidor backend esté fuera de servicio notificarlo en el encabezado.

Notificar correctamente cuando el usuario introduce datos incorrectos.

Funcionabilidad

No se detectaron errores.

Pruebas de rendimiento (aceptación técnica)

Autenticarse: se procede a autenticarse

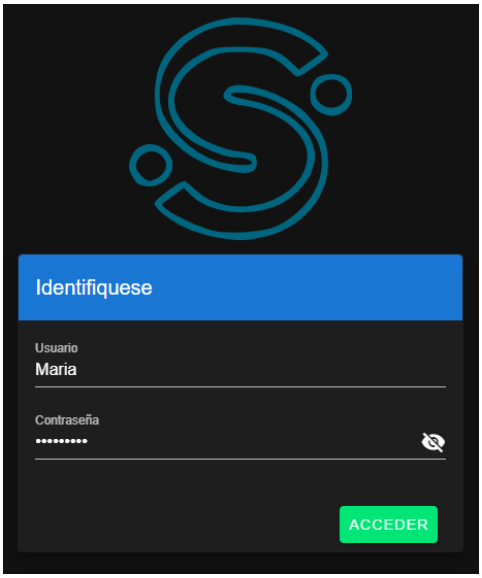


Ilustración 37 Interfaz acceso sub sistema Autenticación (TEST)

Tiempo de respuesta en 1117ms.

Ilustración 38 tiempo de respuesta acceso (TEST)

FORMATO PLAN DE PRUEBAS DE USUARIO			
Identificador	PSA01	**Versión**	V01
Responsable	Ing. Julio Companioni Martínez		
Nombre del caso de prueba	Autenticación		
Módulo		**Sub módulo**	
Autenticación			
Formulario			
/			
Descripción de la prueba			
Rellenar formulario de entrada Acceder			
Resultados esperados			
Autenticación exitosa			
Resultados reales			
Mostrar panel del sistema			
Error			
Imagen			

Tabla 10 Prueba PSA01

Fase 4

Se precede a implementar el sub sistema terminado en un ambiente de producción. Como actualmente el Sistema Integral Informático no está prestando servicio no es una complicación ponerlo en producción. En las fases anteriores quedó definido su modelo, su esquema BDD y se procede ahora a realizar la migración de la base de datos y el sub sistema.

Anexo 2: Sub sistema Mensajes

Fase 1

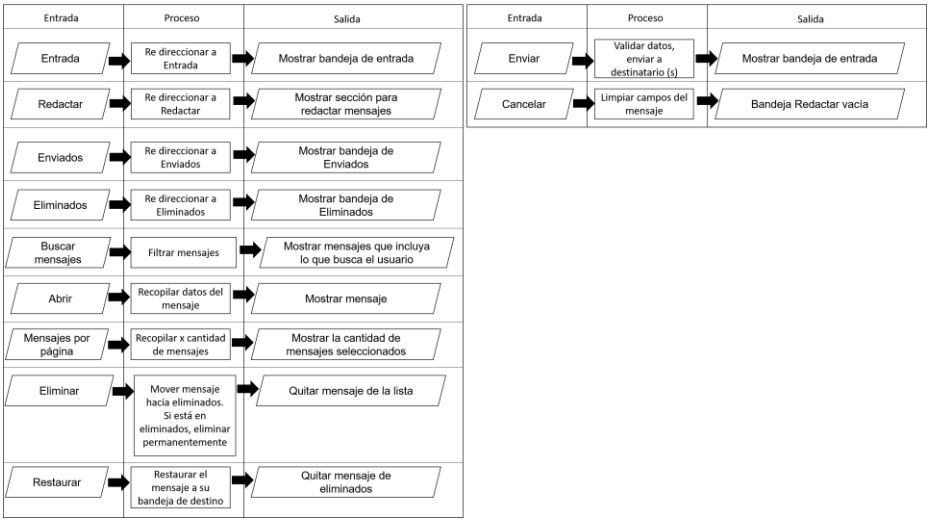


Ilustración 39 Diagrama de entradas y salidas del sub sistema Mensajes

Las dependencias de este sub sistema quedaron reflejadas en el diagrama de entradas y salidas de datos. Estas son:

- Sub sistema Base

Fase 2

Modelado de la base de datos:

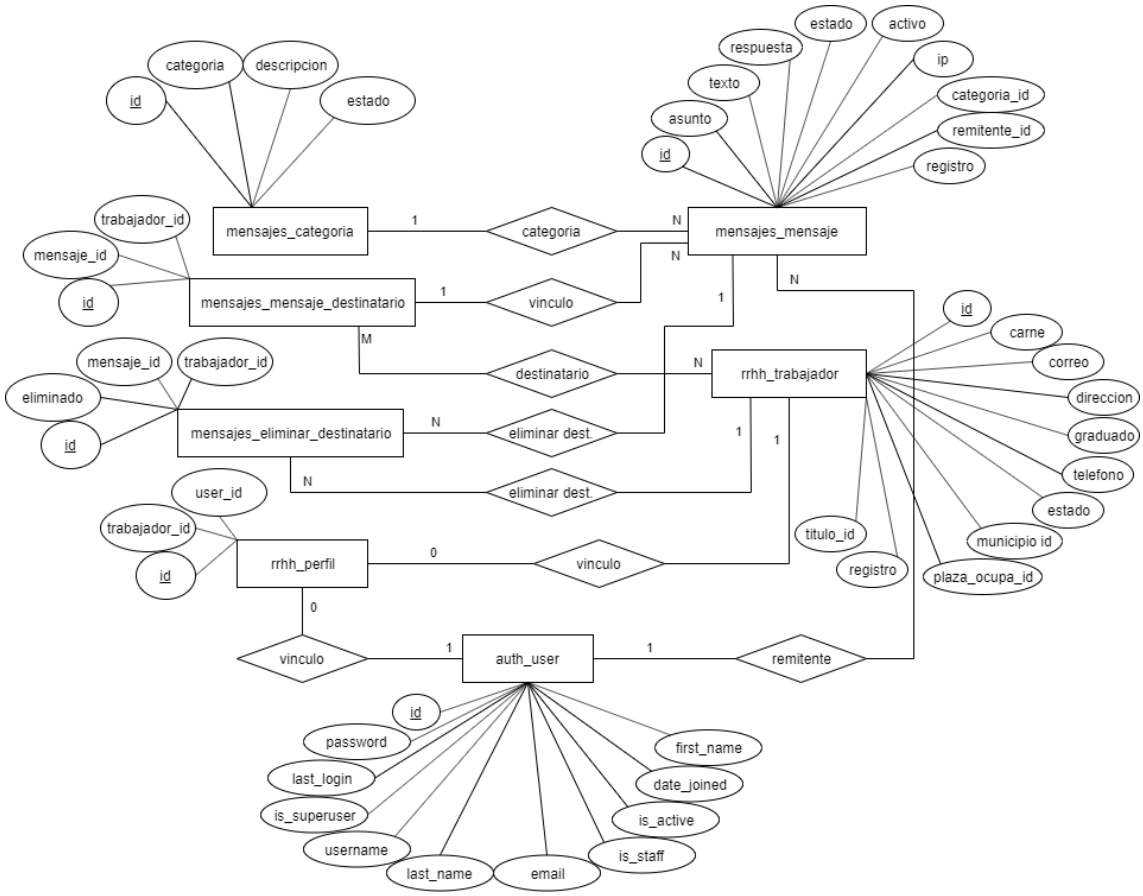


Ilustración 40 Esquema de la base de datos para el sub sistema Mensajes

Interfaces de usuario

Bandeja de entrada. Antiguo/nuevo

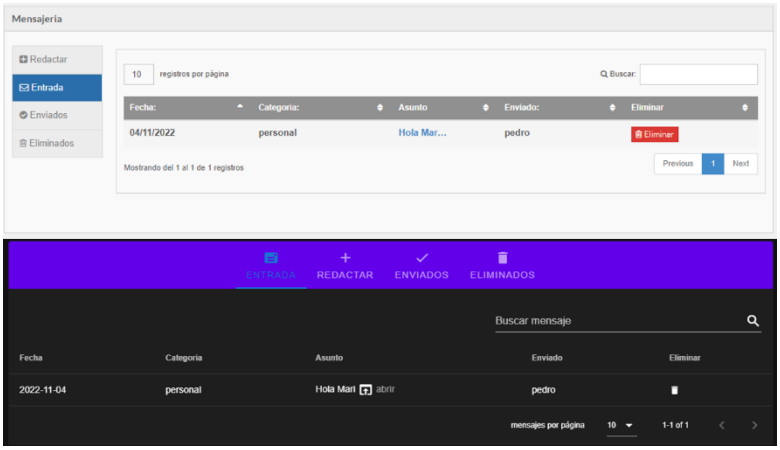


Ilustración 41 Interfaz bandeja de entrada sub sistema Mensajes

Bandeja de enviados. Antiguo/nuevo

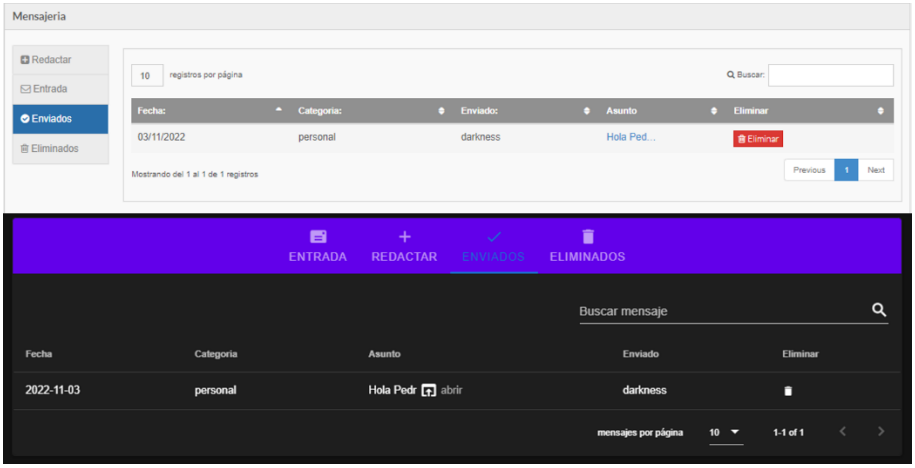


Ilustración 42 Interfaz bandeja de enviados sub sistema Mensajes

Bandeja de eliminados. Antiguo/nuevo

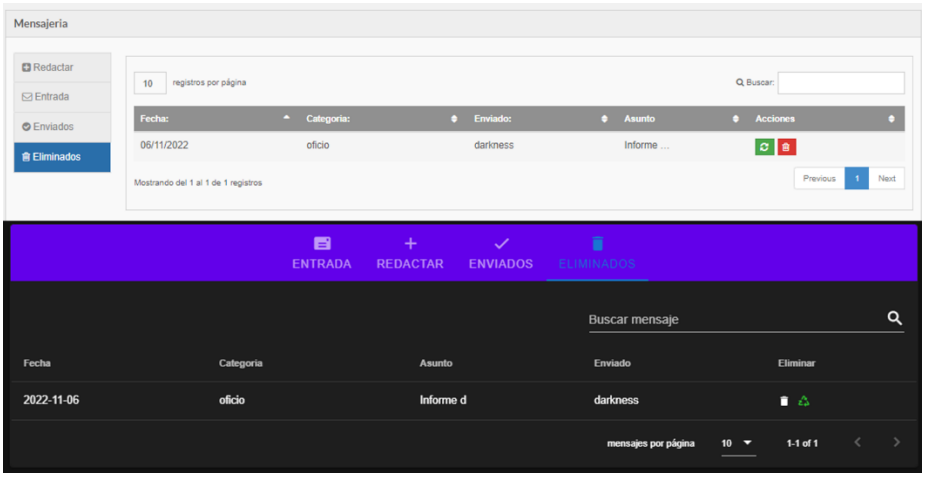


Ilustración 43 Interfaz bandeja de eliminados sub sistema Mensajes

Redactar. Antiguo/nuevo

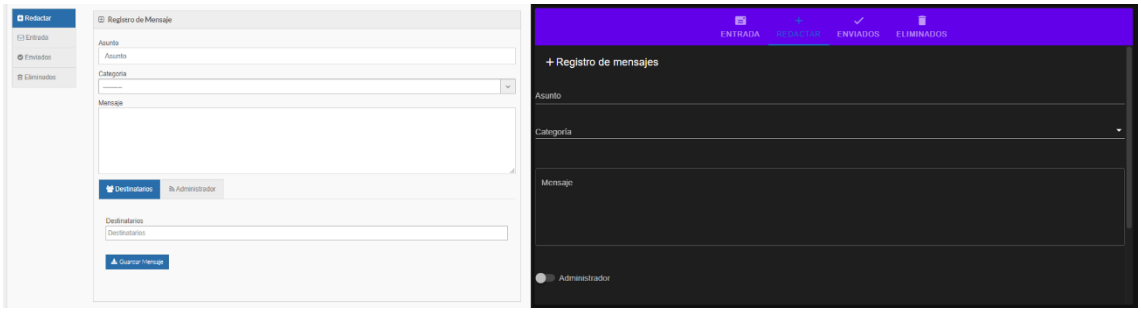


Ilustración 44 Interfaz Redactar sub sistema Mensajes

Detalles mensaje. Antiguo/nuevo

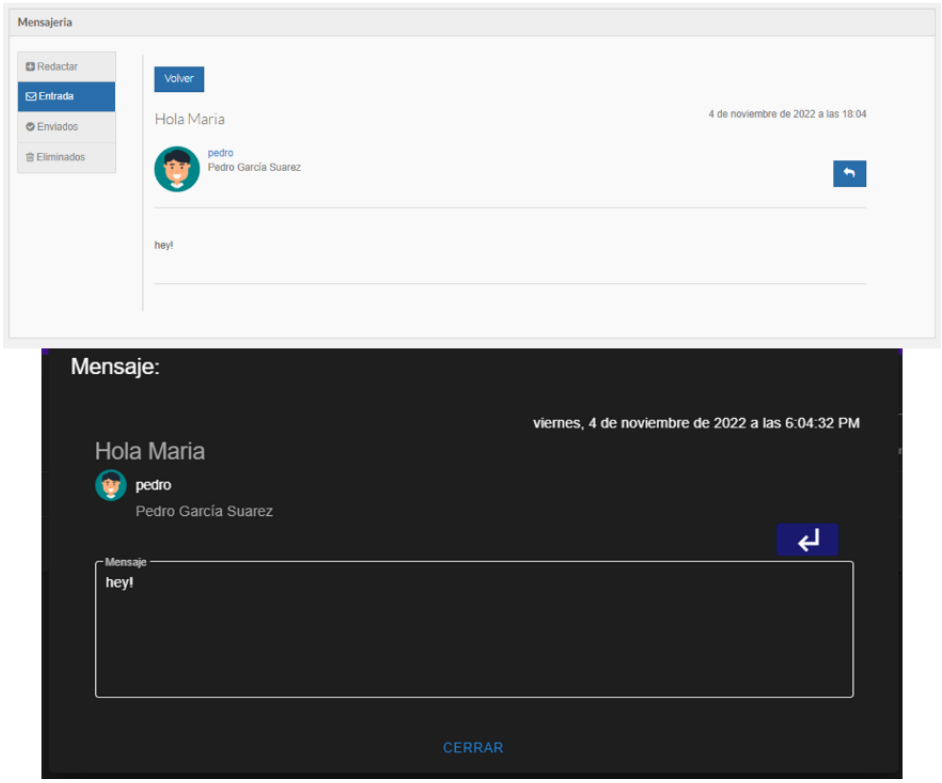


Ilustración 45 Interfaz detalles mensaje sub sistema Mensajes

Comunicación con la interfaz

Backend/Frontend

```python
#elementos para el manejo de las categorias de los mensajes
@api_view(['GET','POST'])
def Categoria_Mensaje_List(request):...

@api_view(['GET','PUT','Delete'])
def Categoria_Mensaje_detail(request, pk):...

#elementor para el manejo de los mensajes
@api_view(['GET','POST'])
def Mensaje_List(request):...

@api_view(['GET','PUT','Delete'])
def Mensaje_detail(request, pk):...

#elementor para el manejo de eliminar destinatarios
@api_view(['GET','POST'])
def Eliminar_Destinatario_List(request):...

@api_view(['GET','PUT','Delete'])
def Eliminar_Destinatario_detail(request, pk):...

#elementor para el manejo de los mensajes leidos
@api_view(['GET','POST'])
def Mensaje_Leido_List(request):...

@api_view(['GET','PUT','Delete'])
def Mensaje_Leido_detail(request, pk):...
```

```javascript
methods: {
  async initialize() {...
  },

  validate () {  ...
  },
  sendData() {...
  },

  deleteItem (item) {...
  },

  deleteItem_permanent (item) {...
  },

  deleteItem_env (item) {...
  },

  restaurarItem (item) {...
  },

  async restaurar(){...
  },

  restDialogLoading(){...
  },

  abrirSMS (item) {...
  },

  abrirSMS_env (item) {...
  },

  async deleteItemConfirm () {...
  },

  SMSItemConfirm () {...
  },

  deleteItemConfirm_permanent() {...
  },

  deleteItemConfirm_env () {...
  },

  SMSItemConfirm () {   ...
  },

  closeDelete () {...
  },

  closeSMS () {...
  },

  async responderSMS(){...
  },

  form_reenvio(enviado, asunto, trabajador){...
  },

  reenvio_default(url){...
  },

  limpiar_envio(){...
  }
```

Ilustración 46 backend/frontend sub sistema Mensajes

Comportamiento del sub sistema con BDD

```
# sub sistema SMS
Feature: Mensajes
    Scenario: Encabezado
    Given Navegador en la pantalla de mensajes del Sistema integral Informatico asumiendo que este autenticado
    When Cuando el usuario  Toque alguna función del encabezado
    Then Mostrar la sección seleccionada por el usuario

    Scenario: eliminar mensajes
    Given Navegador en la pantalla de mensajes en las sección de Entrada, Enviados o Eliminados del Sistema integral
    Informatico asumiendo que este autenticado
    When Cuando el usuario  Toque el icono de eliminar
    Then Mover el mensaje seleccionado a la sección de Eliminados

    Scenario: restaurar mensajes
    Given Navegador en la pantalla de mensajes en las sección de Eliminados del Sistema integral Informatico asumiendo
    que este autenticado
    When Cuando el usuario  Toque el icono de restaurar
    Then Mover el mensaje seleccionado a la sección de Origen

    Scenario: abrir mensajes
    Given Navegador en la pantalla de mensajes en las sección de Entrada o Enviados del Sistema integral Informatico
    asumiendo que este autenticado
    When Cuando el usuario  Toque el icono de abrir
    Then Mostrar información del mensaje seleccionado en un dialogo en pantalla

    Scenario: enviar mensajes
    Given Navegador en la pantalla de mensajes en las sección de Redactar del Sistema integral Informatico asumiendo
    que este autenticado
    When Cuando el usuario  Rellene el formulario y proceda a oprimir enviar
    Then Enviar mensaje a destinatario(s)
    And Recargar y mostrar Entrada
```

Ilustración 47 Comportamiento del sub sistema Mensajes con BDD

Fase 3

Pruebas de funcionalidad (aceptación del usuario)

Tomado Bandeja de entrada como muestra para el Test plan pruebas

Realizado Por: Ing. Julio Companioni Martínez

Ambiente de revisión: TEST

Detalles de la aplicación:

Nombre: Sistema Integral Informático

Plataforma: Django 4.0 + Nuxt.js

Base de datos: MYSQL

Desarrollado Por: Franco Salgado

Opciones de revisión

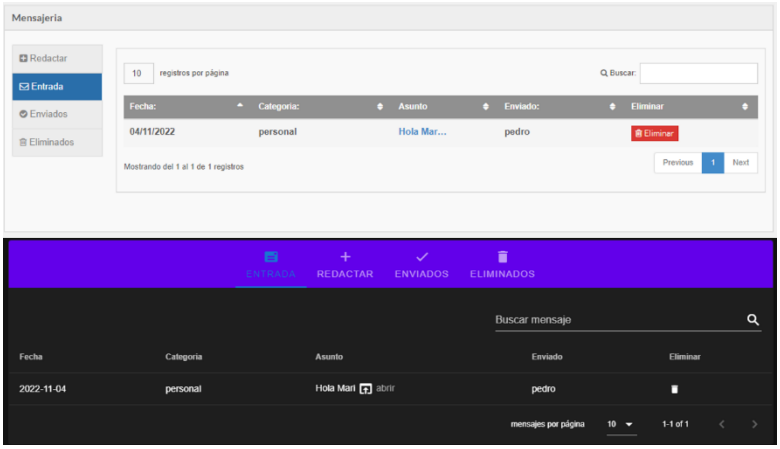


Ilustración 48 Interfaz bandeja de entrada sub sistema Mensajes(TEST)

Opción Bandeja de entrada

Insertar	**Correcto**
Actualizar	Correcto
Eliminar	No tiene

Tabla 11 Opción bandeja de entrada Mensajes (TEST)

Visualización

Cambiar la cantidad de mensajes a mostrar por página para evitar un desborde

Cambiar diseño de la barra de desplazamiento general del sistema

Validación

Un campo de texto que solo contenga un espacio no puede ver válido

Funcionabilidad

Gestionar los mensajes sin necesidad de recargar la página

Pruebas de rendimiento (aceptación técnica)

Enviar mensaje: se procede a enviar un mensaje

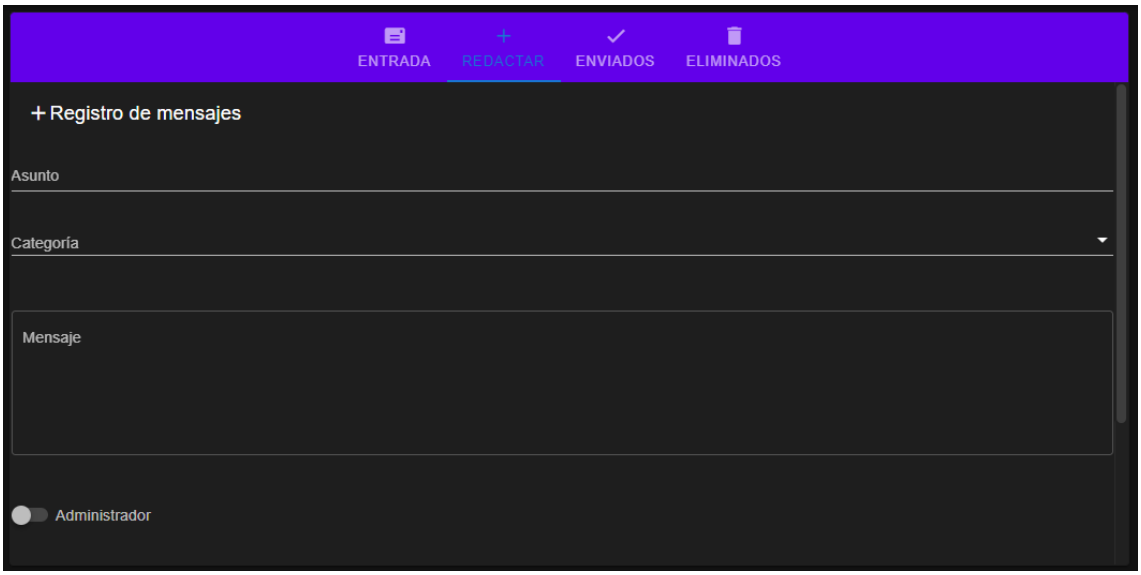


Ilustración 49 Interfaz redactar mensaje sub sistema Mensajes (TEST)

Tiempo de respuesta en 444ms.

Ilustración 50 tiempo de respuesta redactar mensaje(TEST)

FORMATO PLAN DE PRUEBAS DE USUARIO			
Identificador	PSM01	**Versión**	V01
Responsable	Ing. Julio Companioni Martínez		
Nombre del caso de prueba	Enviar mensaje		
Módulo		**Sub módulo**	
Mensajes			
Formulario			
/Mensajes/contacto			
Descripción de la prueba			
Rellenar formulario de envío Enviar			
Resultados esperados			
Mensaje enviado			
Resultados reales			
Mostrar bandeja de entrada			
Error			
Imagen			

Tabla 12 Prueba PSM01

Fase 4

Se precede a implementar el sub sistema terminado en un ambiente de producción. Como actualmente el Sistema Integral Informático no está prestando servicio no es una complicación ponerlo en producción. En las fases anteriores quedó definido su modelo, su esquema BDD y se procede ahora a realizar la migración de la base de datos y el sub sistema.

Anexo 3: Sub sistema Enlaces

Fase 1

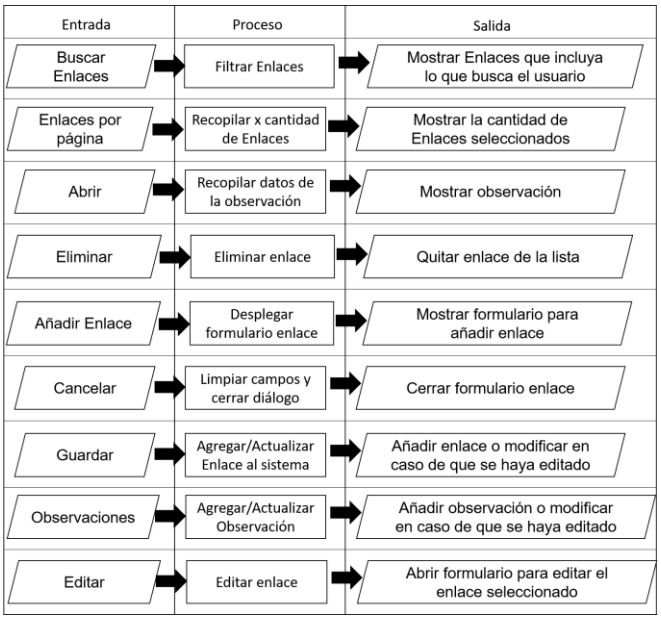


Ilustración 51 Diagrama de entradas y salidas del sub sistema Enlaces

Las dependencias de este sub sistema quedaron reflejadas en el diagrama de entradas y salidas de datos. Estas son:

- Sub sistema Base

Nuevos requerimientos del usuario:

- CRUD Enlaces
- Validaciones correspondientes
- Velocidades en Kb/s y Mb/s

Fase 2

Modelado de la base de datos:

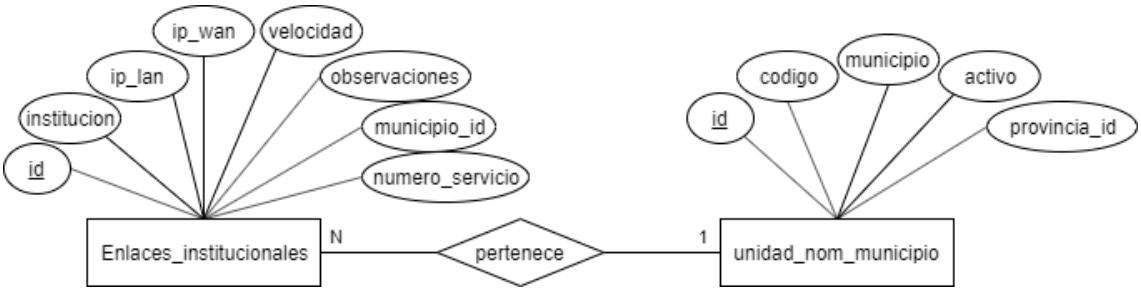


Ilustración 52 Esquema de la base de datos para el sub sistema Enlaces

Interfaces de usuario

Bandeja de enlaces. Antiguo/nuevo

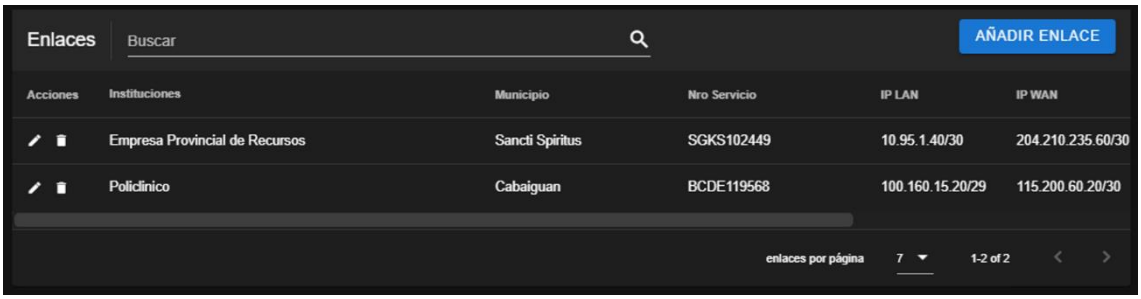


Ilustración 53 Interfaz bandeja de enlaces sub sistema Enlaces

Formulario enlace. Antiguo/nuevo

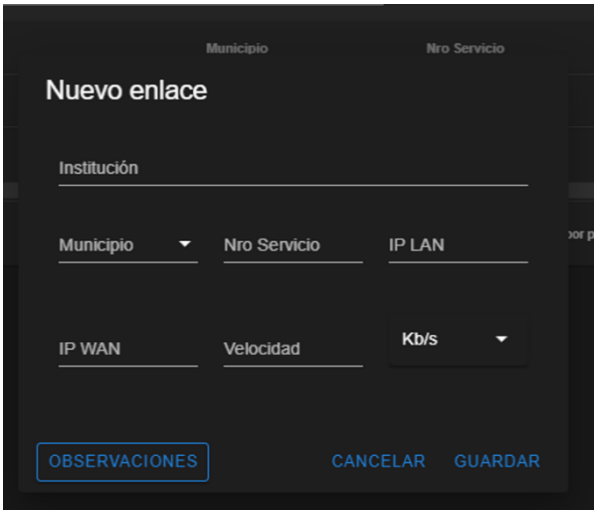


Ilustración 54 Interfaz formulario enlaces sub sistema Enlaces

Comunicación con la interfaz

Backend/Frontend

```python
@api_view(['GET','POST'])

def Enlace_List(request):

    if request.method =='GET':
        enlaces=models.Enlaces_Institucionales.objects.all()
        serializer= serializers.EnlaceListSerializer(enlaces,many=True)
        return Response(serializer.data)

    elif request.method =='POST':
        serializer= serializers.EnlacePutPostSerializer(data=request.data)

        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)


@api_view(['GET','PUT','Delete'])
def Enlace_detail(request, pk):···
```

```javascript
async initialize() {···
},

validate () {if(this.$refs.form.validate()){this.save()}},
validate_observer () {if(this.$refs.form_observer.validate()){this.close_dialogObser()}},

editItem(item){···
},

deleteItem (item) {···
},

deleteItemConfirm () {···
},

close () {···
},

dialogObser(){···
},

close_dialogObser(){···
},

closeDelete () {···
},
institucion_full(item){this.full_institucion=item.institucion},
clear_inst(){this.full_institucion=''},

save () {···
},

abrirObs (item) {···
},
```

Ilustración 55 backend/frontend sub sistema Enlaces

Comportamiento del sub sistema con BDD

```
# sub sistema Enlaces
Feature: Enlaces Institucionales
    Scenario: formulario enlace
    Given Navegador en la pantalla de Enlaces del Sistema integral Informatico asumiendo que este autenticado
    When Cuando el usario  Toque la función añadir enlace o editar enlace
    Then Mostrar formulario enlace

    Scenario: añadir enlace
    Given Navegador en la pantalla de Enlaces con el formulario enlace desplegado del Sistema integral Informatico
    asumiendo que este autenticado
    When Cuando el usario  Toque la función Guardar
    Then Añadir enlace al sistema
    And Cerrar formulario enlace

    Scenario: editar enlace
    Given Navegador en la pantalla de Enlaces con el formulario enlace desplegado del Sistema integral Informatico asumiendo
    que este autenticado
    When Cuando el usario  Toque la función Guardar
    Then actualizar enlace seleccionado al sistema
    And Cerrar formulario enlace

    Scenario: eliminar enlace
    Given Navegador en la pantalla de Enlaces del Sistema integral Informatico asumiendo que este autenticado
    When Cuando el usario  Toque la función eliminar enlace
    Then Mostrar Dialogo de confirmar eliminar enlace
    And con la confirmación proceder a eliminar el enlace del sistema
```

Ilustración 56 Comportamiento del sub sistema Enlaces con BDD

Fase 3

Pruebas de funcionalidad (aceptación del usuario)

Tomado Bandeja de enlaces como muestra para el Test plan pruebas

Realizado Por: Ing. Julio Companioni Martínez

Ambiente de revisión: TEST

Detalles de la aplicación:

Nombre: Sistema Integral Informático

Plataforma: Django 4.0 + Nuxt.js

Base de datos: MYSQL

Desarrollado Por: Franco Salgado

Opciones de revisión

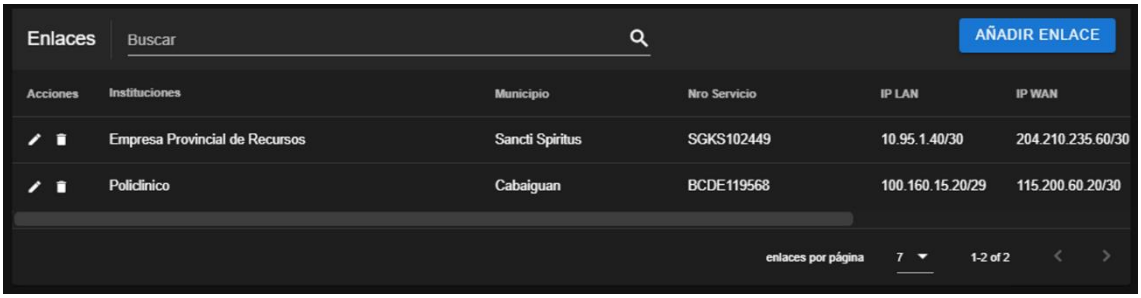


Ilustración 57 Interfaz bandeja de enlaces sub sistema Enlaces(TEST)

Opción Bandeja de enlaces

Insertar	**Correcto**
Actualizar	Correcto
Eliminar	Correcto

Tabla 13 Opción bandeja de enlaces Enlaces(TEST)

Visualización

No se detectaron errores

Validación

Validar que cada ip sea exclusivamente único.

Funcionabilidad

Remover ip de elemento editado actualmente para que no interfiera la validación de ip único, si cancela incorporarlos de nuevo

Pruebas de rendimiento (aceptación técnica)

Añadir enlace: se procede a añadir enlace

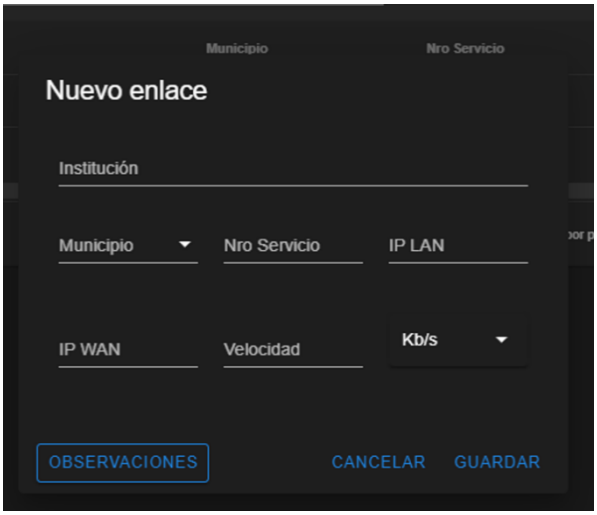


Ilustración 58 Interfaz añadir enlace sub sistema Mensajes (TEST)

Tiempo de respuesta en 467ms.

Ilustración 59 tiempo de respuesta añadir enlace (TEST)

FORMATO PLAN DE PRUEBAS DE USUARIO			
Identificador	PSE01	**Versión**	V01
Responsable	Ing. Julio Companioni Martínez		
Nombre del caso de prueba	Añadir enlace		
Módulo		**Sub módulo**	
Enlace			
Formulario			
/enlaces			
Descripción de la prueba			
Rellenar formulario enlace Guardar			
Resultados esperados			
Agregado exitosamente			
Resultados reales			
Cerrar dialogo y mostrar enlace creado			
Error			
Imagen			

Tabla 14 Prueba PSE01

Fase 4

Se precede a implementar el sub sistema terminado en un ambiente de producción. Como actualmente el Sistema Integral Informático no está prestando servicio no es una complicación ponerlo en producción. En las fases anteriores quedó definido su modelo, su esquema BDD y se procede ahora a realizar la migración de la base de datos y el sub sistema.

Anexo 4: Sub sistema Departamento

Fase 1

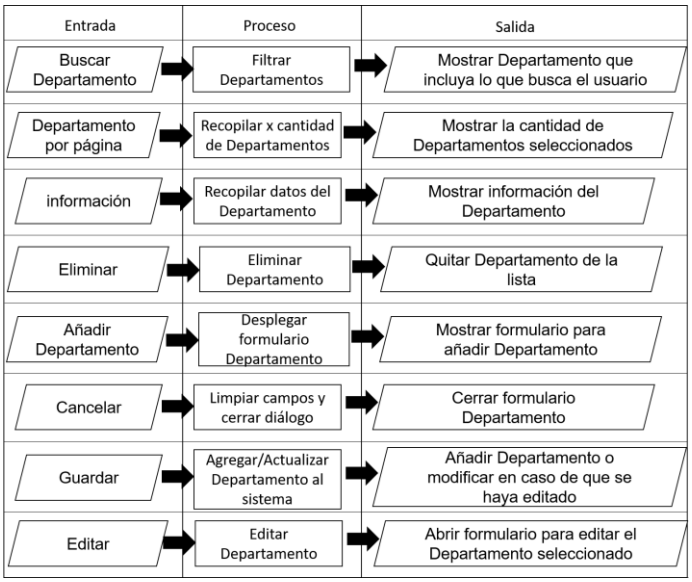


Ilustración 60 Diagrama de entradas y salidas del sub sistema Departamento

Las dependencias de este sub sistema quedaron reflejadas en el diagrama de entradas y salidas de datos. Estas son:

- Sub sistema Base.

Nuevos requerimientos del usuario

- Filtrar Departamentos.

Fase 2

Modelado de la base de datos:

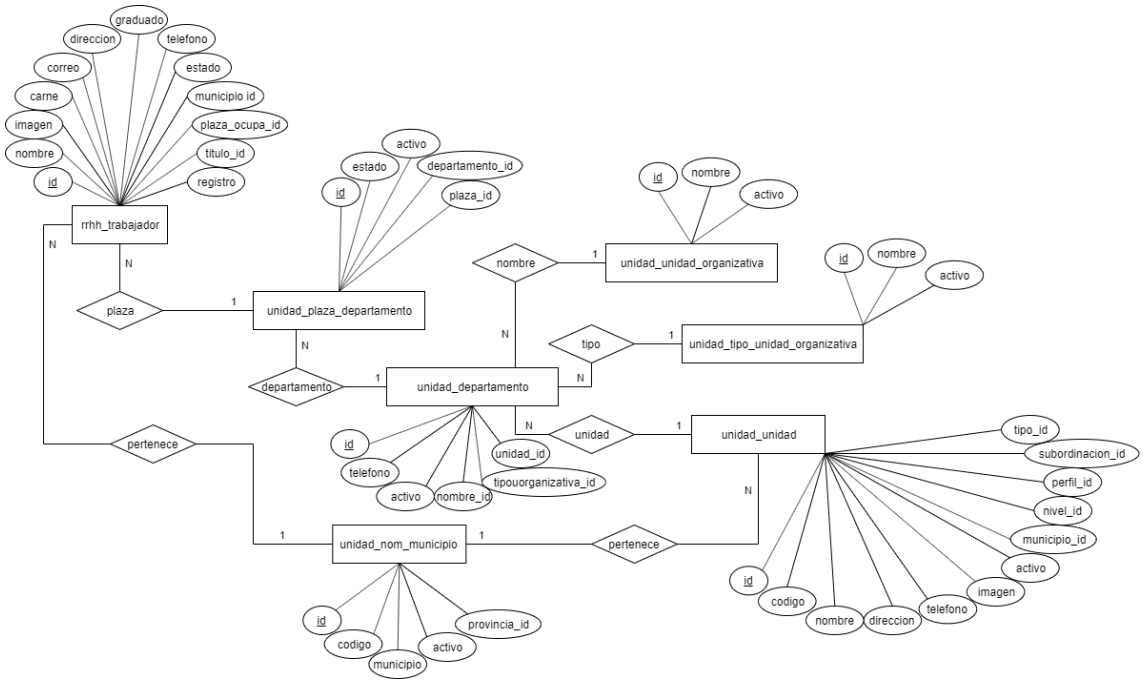


Ilustración 61 Esquema de la base de datos para el sub sistema Departamento

Interfaces de usuario

Bandeja de Departamentos. Antiguo/nuevo

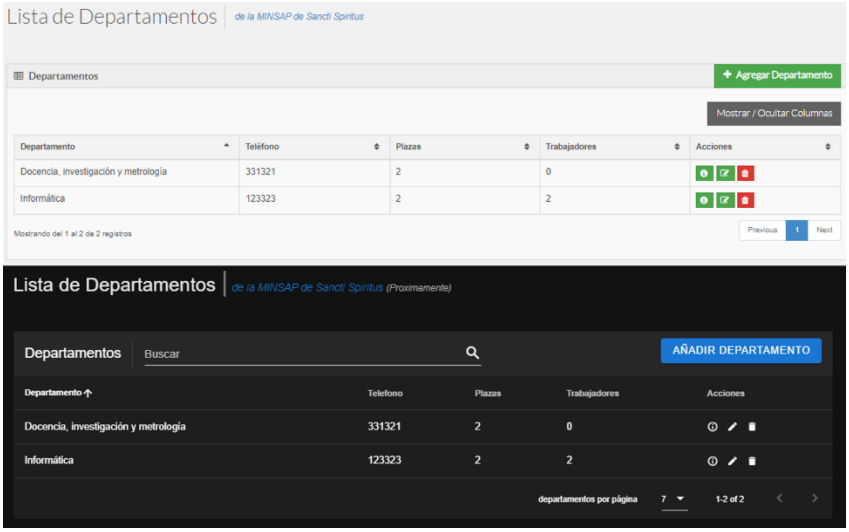


Ilustración 62 Interfaz bandeja de departamentos sub sistema Departamento

Formulario Departamento. Antiguo/nuevo

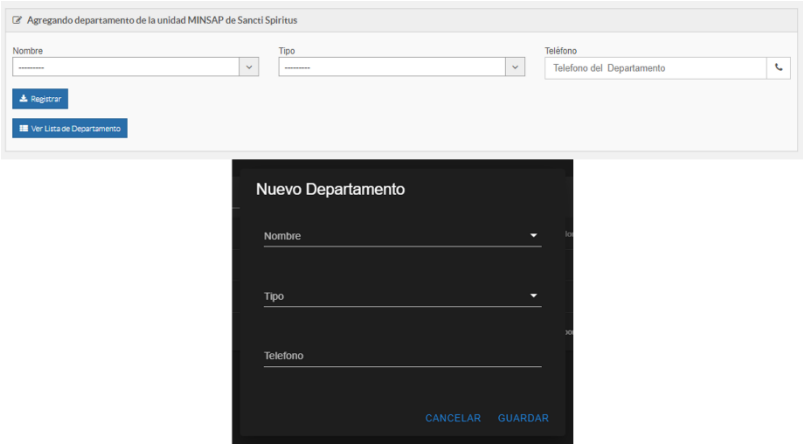


Ilustración 63 Interfaz formulario departamento sub sistema Departamento

Información Departamento. Antiguo/nuevo

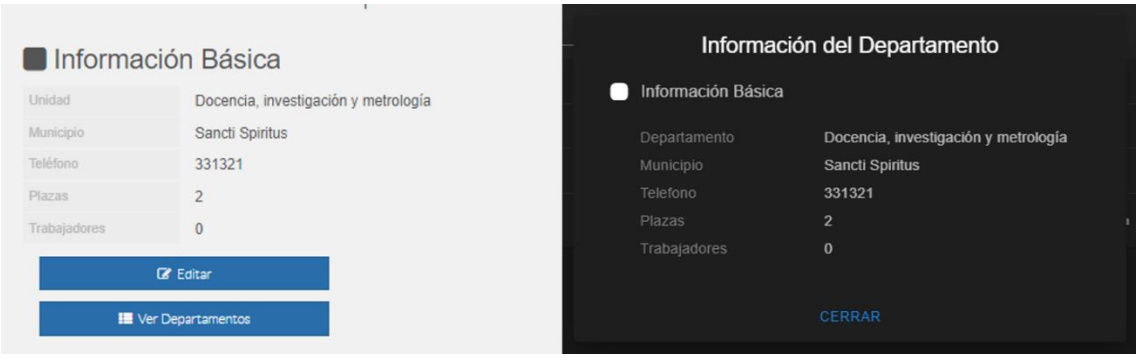


Ilustración 64 Interfaz información departamento sub sistema Departamento

Comunicación con la interfaz

Backend/Frontend

```python
# elementos del trabajador en rrhh

class TrabajadorGAPIViewList(generics.GenericAPIView, m

class TrabajadorGAPIViewDetail(generics.GenericAPIView,

# Nombre de tipo de unidad organizativa
@api_view(['GET','POST'])
def Tipo_Unidad_Org_List(request):...

@api_view(['GET','PUT','Delete'])
def Tipo_Unidad_Org_detail(request, pk):...

# add unidad organizativa

@api_view(['GET','POST'])
def Unidad_Org_List(request):...

@api_view(['GET','PUT','Delete'])
def Unidad_Org_detail(request, pk):...

# Municipio

@api_view(['GET','POST'])
def Municipio_List(request):...

@api_view(['GET','PUT','Delete'])
def Municipio_detail(request, pk):...

# add Unidad
class UnidadGAPIViewList(generics.GenericAPIView, m

class UnidadGAPIViewDetail(generics.GenericAPIView

# add Departamento

@api_view(['GET','POST'])
def Departamento_List(request):...

@api_view(['GET','PUT','Delete'])
def Departamento_detail(request, pk):...

# add Plaza de Departamento

@api_view(['GET','POST'])
def Plaza_Departamento_List(request):...

@api_view(['GET','PUT','Delete'])
def Plaza_Departamento_detail(request, pk):...
```

```javascript
async initialize() {...
  },

editItem (item) {...
},

deleteItem (item) {...
},

async deleteItemConfirm () {...
},

close () {...
},

closeDelete () {...
},

reload(){...
},
validate () {if(this.$refs.form.validate()){this.save()}},

save () {...
},

async info_departamento(item){...
},
```

Ilustración 65 backend/frontend sub sistema Departamento

Comportamiento del sub sistema con BDD

```
# sub sistema Dpto
Feature: Departamentos
    Scenario: formulario Departamento
    Given Navegador en la pantalla de Departamentos del Sistema integral Informatico asumiendo que este autenticado
    When Cuando el usario  Toque la función añadir Departamento o editar Departamento
    Then Mostrar formulario Departamento

    Scenario: añadir Departamento
    Given Navegador en la pantalla de Departamentos con el formulario Departamento desplegado del Sistema integral
    Informatico asumiendo que este autenticado
    When Cuando el usario  Toque la función Guardar
    Then Añadir Departamento al sistema
    And Cerrar formulario Departamento

    Scenario: editar Departamento
    Given Navegador en la pantalla de Departamentos con el formulario Departamento desplegado del Sistema integral
    Informatico asumiendo que este autenticado
    When Cuando el usario  Toque la función Guardar
    Then actualizar Departamento seleccionado al sistema
    And Cerrar formulario Departamento

    Scenario: eliminar Departamento
    Given Navegador en la pantalla de Departamentos del Sistema integral Informatico asumiendo que este autenticado
    When Cuando el usario  Toque la función eliminar Departamento
    Then Mostrar Dialogo de confirmar eliminar Departamento
    And con la confirmación proceder a eliminar el Departamento del sistema

    Scenario: información Departamento
    Given Navegador en la pantalla de Departamentos del Sistema integral Informatico asumiendo que este autenticado
    When Cuando el usario  Toque la función información del Departamento
    Then Mostrar Dialogo con los datos del Departamento
```

Ilustración 66 Comportamiento del sub sistema Departamento con BDD

Fase 3

Pruebas de funcionalidad (aceptación del usuario)

Tomado Bandeja Departamentos como muestra para el Test plan pruebas

Realizado Por: Ing. Julio Companioni Martínez

Ambiente de revisión: TEST

Detalles de la aplicación:

Nombre: Sistema Integral Informático

Plataforma: Django 4.0 + Nuxt.js

Base de datos: MYSQL

Desarrollado Por: Franco Salgado

Opciones de revisión

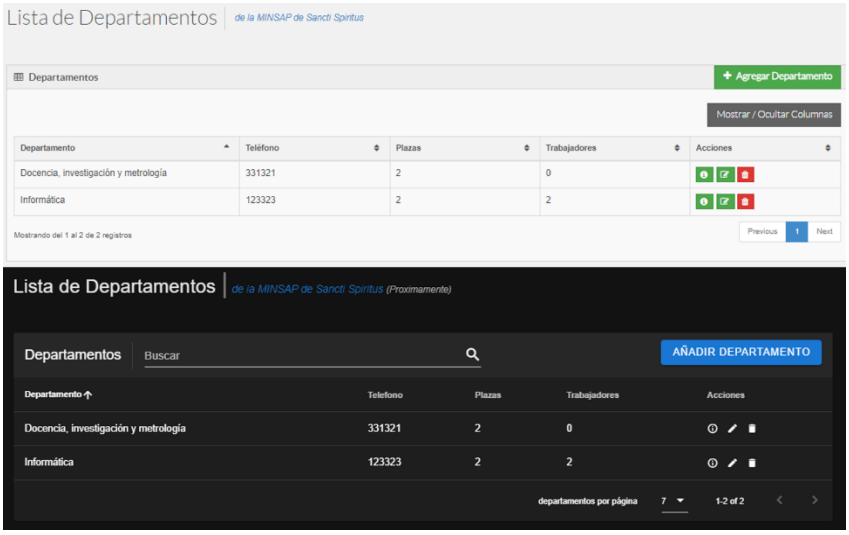


Ilustración 67 Interfaz bandeja de departamentos sub sistema Departamento(TEST)

Opción Bandeja de departamentos

Insertar	**Correcto**
Actualizar	Correcto
Eliminar	Correcto

Tabla 15 Opción bandeja de departamentos Departamento(TEST)

Visualización

No se detectaron errores.

Validación

No se detectaron errores.

Funcionabilidad

Al abrir y cerrar dos veces añadir departamento, a la segunda no se cierra el formulario Departamento.

Pruebas de rendimiento (aceptación técnica)

Añadir Departamento: se procede a añadir un departamento

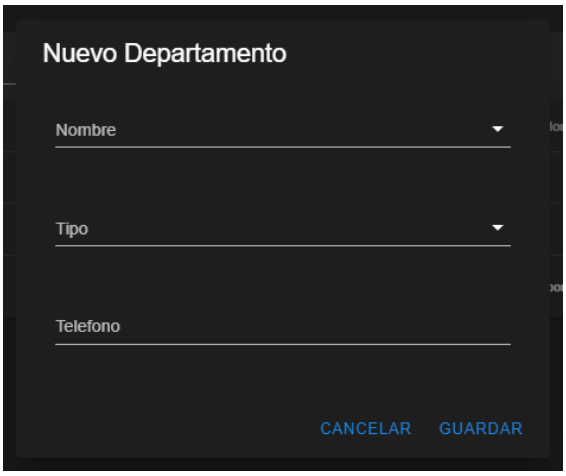


Ilustración 68 Interfaz añadir departamento sub sistema Departamento(TEST)

Tiempo de respuesta en 278ms.

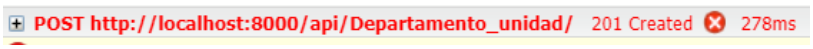


Ilustración 69 tiempo de respuesta añadir departamento(TEST)

FORMATO PLAN DE PRUEBAS DE USUARIO			
Identificador	PSD01	**Versión**	V01
Responsable	Ing. Julio Companioni Martínez		
Nombre del caso de prueba	Añadir departamento		
Módulo		**Sub módulo**	
Departamentos			
Formulario			
/departamentos			
Descripción de la prueba			
Rellenar formulario Departamento Guardar			
Resultados esperados			
Departamento añadido			
Resultados reales			
Departamento agregado correctamente			
Error			
Imagen			
			

Tabla 16 Prueba PSD01

Fase 4

Se precede a implementar el sub sistema terminado en un ambiente de producción. Como actualmente el Sistema Integral Informático no está prestando servicio no es una complicación ponerlo en producción. En las fases anteriores quedó definido su modelo, su

esquema BDD y se procede ahora a realizar la migración de la base de datos y el sub sistema.

Anexo 5: Sub sistema Plazas

Fase 1

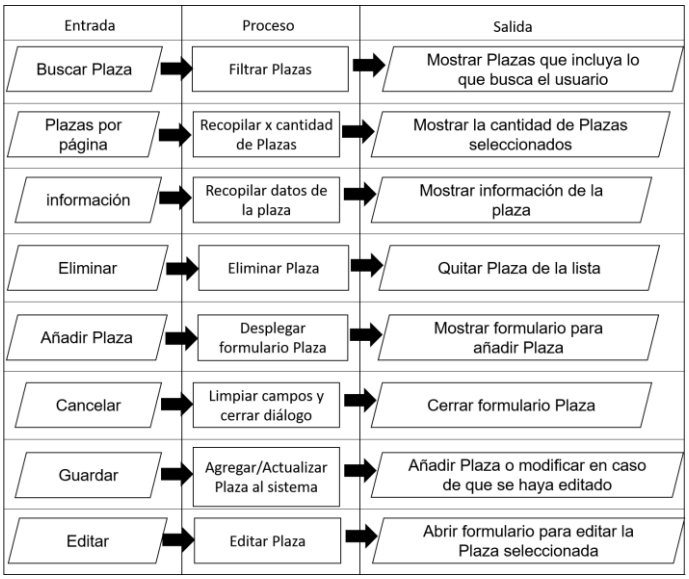


Ilustración 70 Diagrama de entradas y salidas del sub sistema Plazas

Las dependencias de este sub sistema quedaron reflejadas en el diagrama de entradas y salidas de datos. Estas son:

- Sub sistema Base
- Sub sistema Departamento

Nuevos requerimientos del usuario:

- Filtrar plazas.

Fase 2

Modelado de la base de datos:

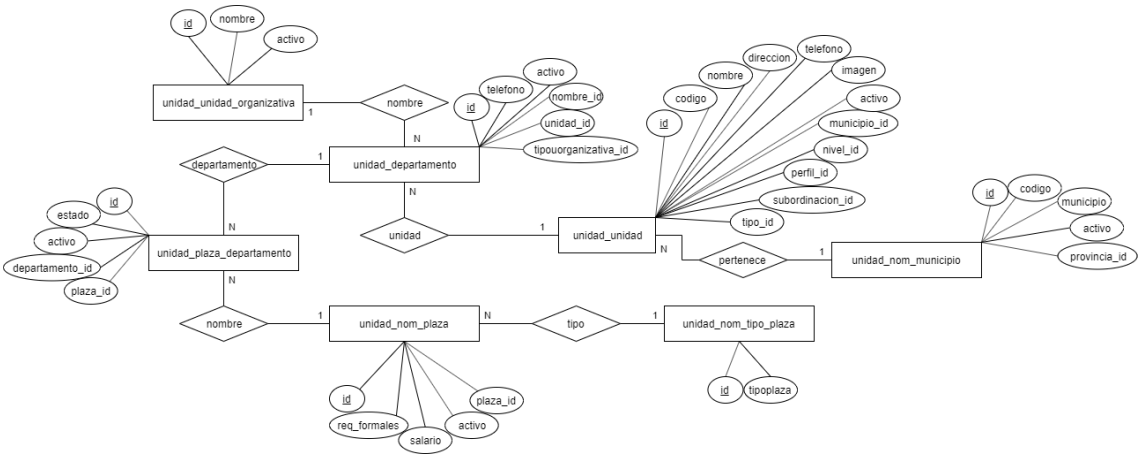


Ilustración 71 Esquema de la base de datos para el sub sistema Plazas

Interfaces de usuario

Bandeja de Plazas. Antiguo/nuevo

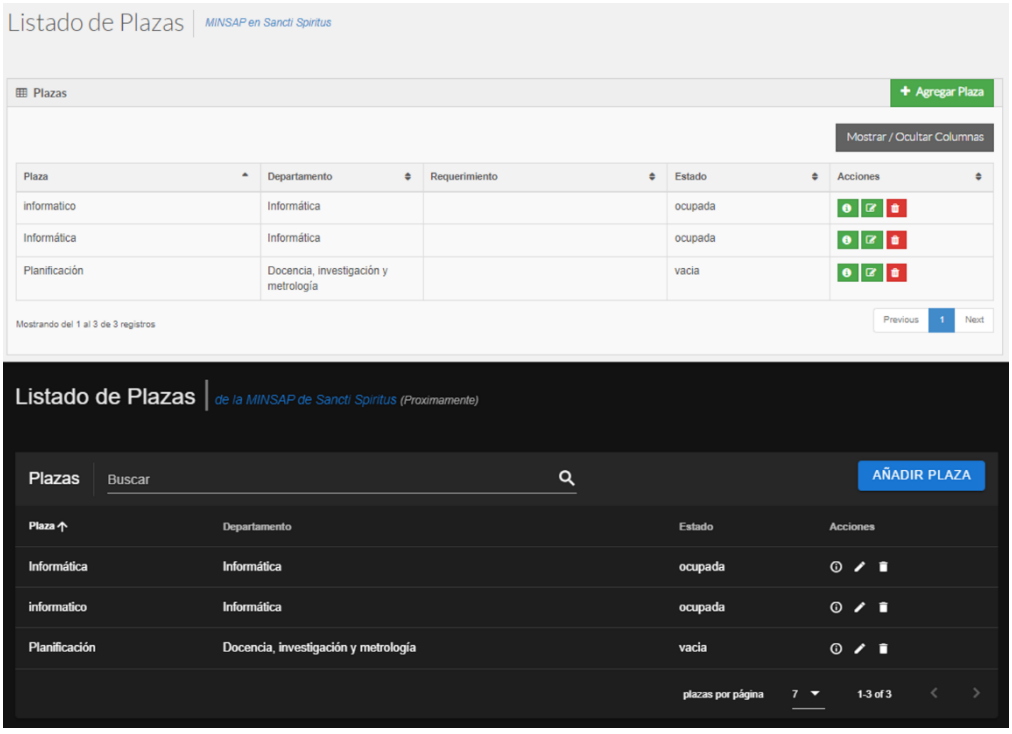


Ilustración 72 Interfaz bandeja de plazas sub sistema Plazas

Formulario Plaza. Antiguo/nuevo

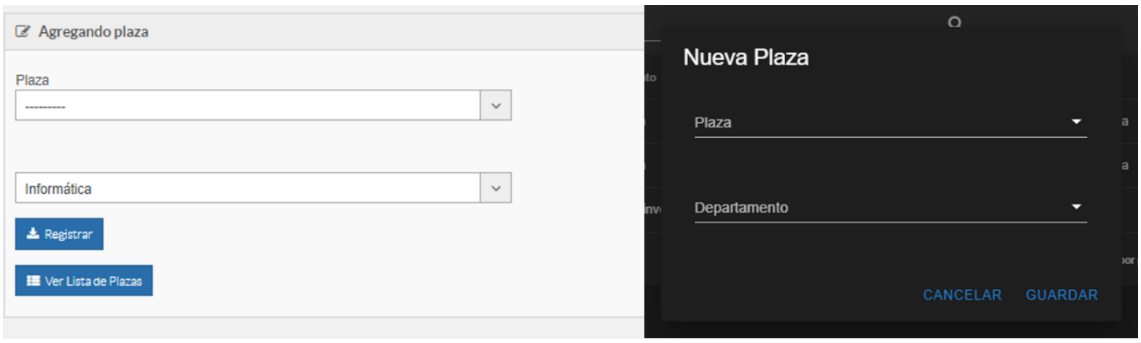


Ilustración 73 Interfaz formulario plaza sub sistema Plazas

Información Plaza. Antiguo/nuevo

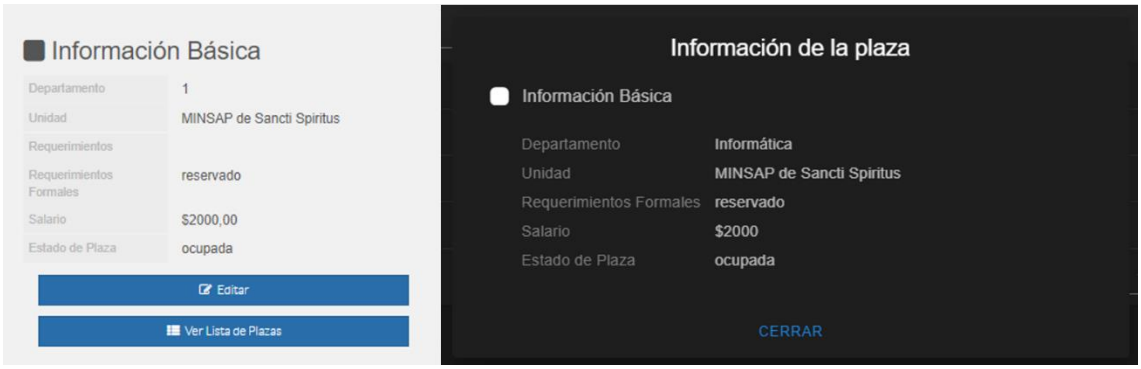


Ilustración 74 Interfaz información plaza sub sistema Plazas

Comunicación con la interfaz

Backend/Frontend

```
# Nombre de plaza de departamento
@api_view(['GET','POST'])
def Nombre_Plaza_List(request):···

@api_view(['GET','PUT','Delete'])
def Nombre_Plaza_detail(request, pk):···

# add unidad organizativa
@api_view(['GET','POST'])
def Unidad_Org_List(request):···

@api_view(['GET','PUT','Delete'])
def Unidad_Org_detail(request, pk):···

# Municipio
@api_view(['GET','POST'])
def Municipio_List(request):···

@api_view(['GET','PUT','Delete'])
def Municipio_detail(request, pk):···

# add Unidad
class UnidadGAPIViewList(generics.GenericAPIView, m


class UnidadGAPIViewDetail(generics.GenericAPIView,


# add Departamento
@api_view(['GET','POST'])
def Departamento_List(request):···

@api_view(['GET','PUT','Delete'])
def Departamento_detail(request, pk):···

# add Plaza de Departamento
@api_view(['GET','POST'])
def Plaza_Departamento_List(request):···

@api_view(['GET','PUT','Delete'])
def Plaza_Departamento_detail(request, pk):···
```

```
  async initialize() {···
  },
  editItem (item) {···
},
deleteItem (item) {···
},

async deleteItemConfirm () {···
},

close () {···
},

closeDelete () {···
},
reload(){···
},
validate () {if(this.$refs.form.validate()){this.save()}},
save () {···
},
async info_plaza(item){···
}
```

Ilustración 75 backend/frontend sub sistema Plazas

Comportamiento del sub sistema con BDD

```
# sub sistema Plaza
Feature: Departamentos
    Scenario: formulario Plaza
    Given Navegador en la pantalla de Plazas del Sistema integral Informatico asumiendo que este autenticado
    When Cuando el usario  Toque la función añadir Plaza o editar Plaza
    Then Mostrar formulario Plaza

    Scenario: añadir Plaza
    Given Navegador en la pantalla de Plazas con el formulario Plaza desplegado del Sistema integral Informatico
    asumiendo que este autenticado
    When Cuando el usario  Toque la función Guardar
    Then Añadir Plaza al sistema
    And Cerrar formulario Plaza

    Scenario: editar Plaza
    Given Navegador en la pantalla de Plazas con el formulario Plaza desplegado del Sistema integral Informatico
    asumiendo que este autenticado
    When Cuando el usario  Toque la función Guardar
    Then actualizar Plaza seleccionado al sistema
    And Cerrar formulario Plaza

    Scenario: eliminar Plaza
    Given Navegador en la pantalla de Plazas del Sistema integral Informatico asumiendo que este autenticado
    When Cuando el usario  Toque la función eliminar Plaza
    Then Mostrar Dialogo de confirmar eliminar Plaza
    And con la confirmación proceder a eliminar la Plaza del sistema

    Scenario: información Plaza
    Given Navegador en la pantalla de Plazas del Sistema integral Informatico asumiendo que este autenticado
    When Cuando el usario  Toque la función información de la Plaza
    Then Mostrar Dialogo con los datos de la Plaza
```

Ilustración 76 Comportamiento del sub sistema Plazas con BDD

Fase 3

Pruebas de funcionalidad (aceptación del usuario)

Tomado Bandeja de plazas como muestra para el Test plan pruebas

Realizado Por: Ing. Julio Companioni Martínez

Ambiente de revisión: TEST

Detalles de la aplicación:

Nombre: Sistema Integral Informático

Plataforma: Django 4.0 + Nuxt.js

Base de datos: MYSQL

Desarrollado Por: Franco Salgado

Opciones de revisión

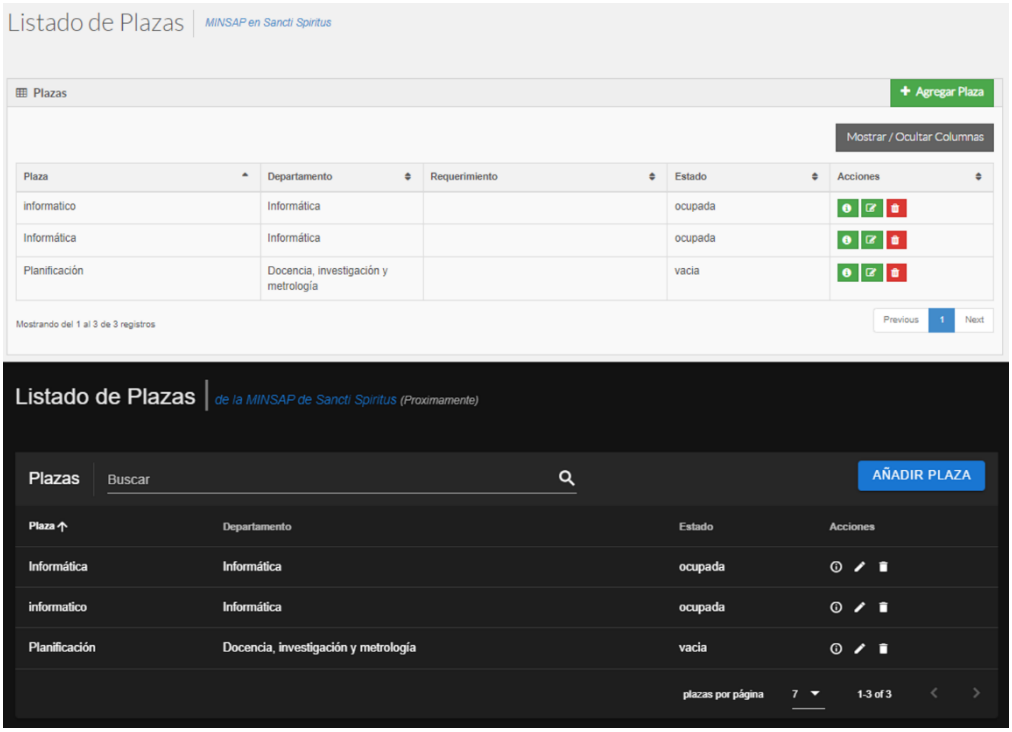


Ilustración 77 Interfaz bandeja de plazas sub sistema Plazas(TEST)

Opción Bandeja de Plazas

Insertar	**Correcto**
Actualizar	Correcto
Eliminar	Correcto

Tabla 17 Opción bandeja de plazas Plazas(TEST)

Visualización

Ordenar las plazas por su nombre por defecto.

Corregir ancho de la tabla.

Poner el buscador más a la izquierda.

Validación

No se detectaron errores.

Funcionabilidad

No se detectaron errores.

Pruebas de rendimiento (aceptación técnica)

Añadir plaza: se procede a añadir una plaza

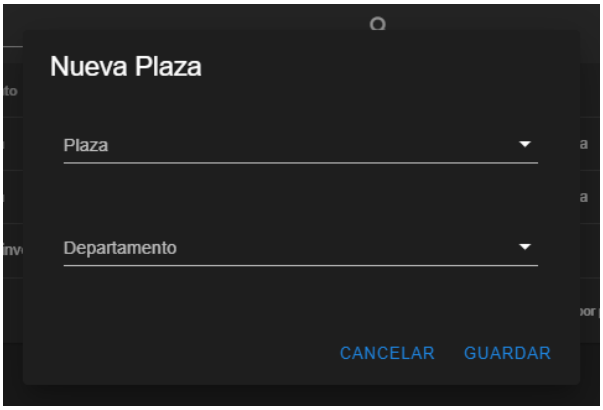


Ilustración 78 Interfaz añadir plaza sub sistema Plazas(TEST)

Tiempo de respuesta en 169ms.

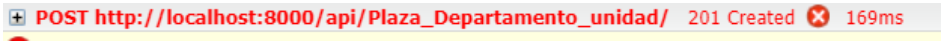


Ilustración 79 tiempo de respuesta añadir plaza(TEST)

FORMATO PLAN DE PRUEBAS DE USUARIO			
Identificador	PSP01	**Versión**	V01
Responsable	Ing. Julio Companioni Martínez		
Nombre del caso de prueba	Añadir plaza		
Módulo		**Sub módulo**	
Plazas			
Formulario			
/plazas			
Descripción de la prueba			
Rellenar formulario Plaza Guardar			
Resultados esperados			
Plaza añadida			
Resultados reales			
Plaza agregada correctamente			
Error			
Imagen			
			

Tabla 18 Prueba PSP01

Fase 4

Se precede a implementar el sub sistema terminado en un ambiente de producción. Como actualmente el Sistema Integral Informático no está prestando servicio no es una complicación ponerlo en producción. En las fases anteriores quedó definido su modelo, su

esquema BDD y se procede ahora a realizar la migración de la base de datos y el sub sistema.

Anexo 6: Sub sistema RRHH

Fase 1

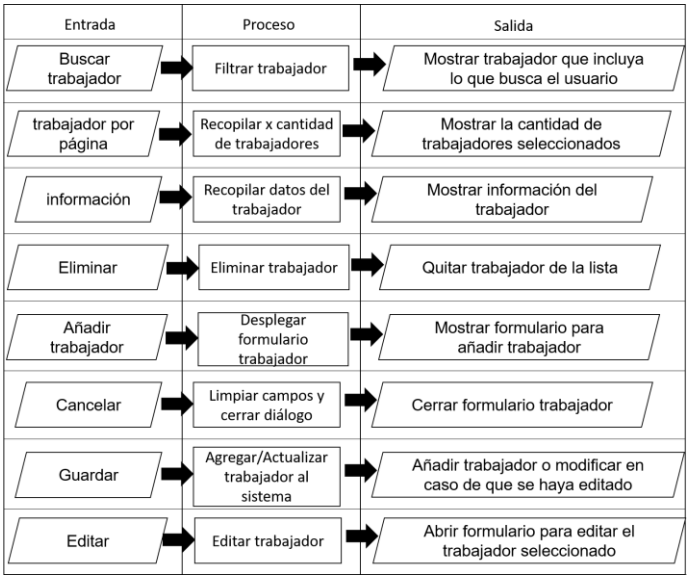


Ilustración 80 Diagrama de entradas y salidas del sub sistema RRHH

Las dependencias de este sub sistema quedaron reflejadas en el diagrama de entradas y salidas de datos. Estas son:

- Sub sistema Base
- Sub sistema Plazas

Nuevos requerimientos del usuario:

- Filtrar trabajadores.

Fase 2

Modelado de la base de datos:

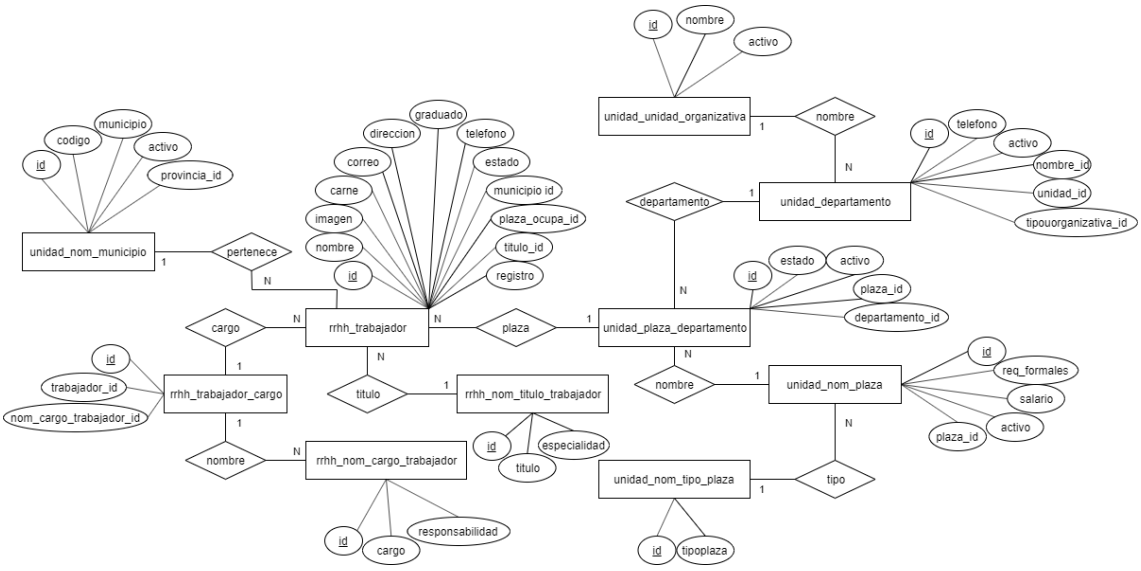


Ilustración 81 Esquema de la base de datos para el sub sistema RRHH

Interfaces de usuario

Bandeja de Trabajadores. Antiguo/nuevo

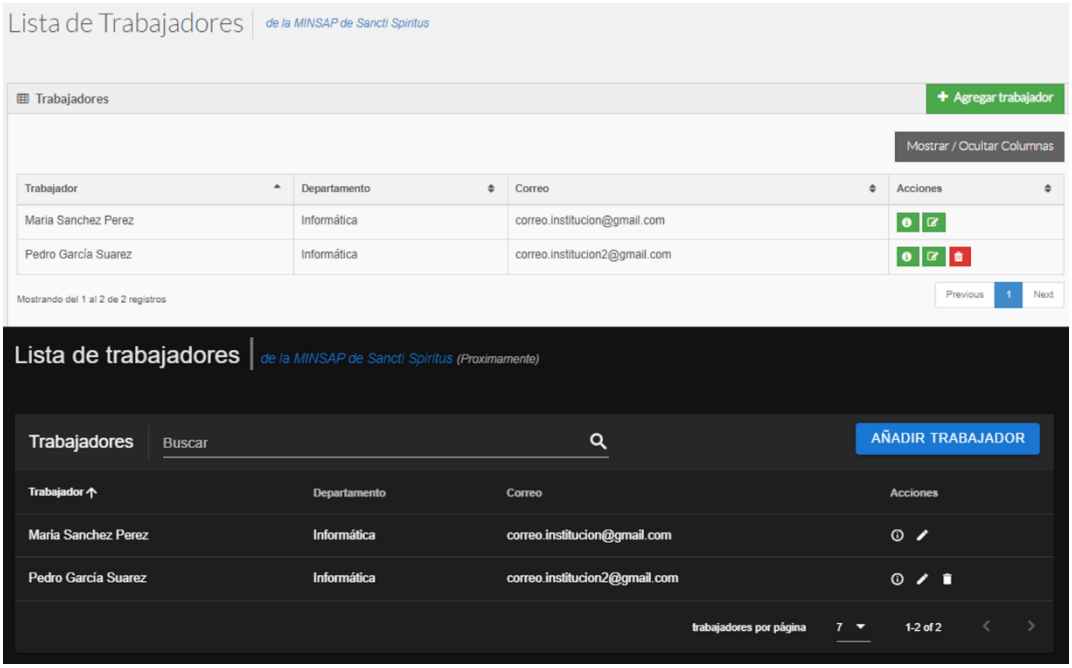


Ilustración 82 Interfaz bandeja de trabajadores sub sistema RRHH

Formulario Trabajador. Antiguo/nuevo

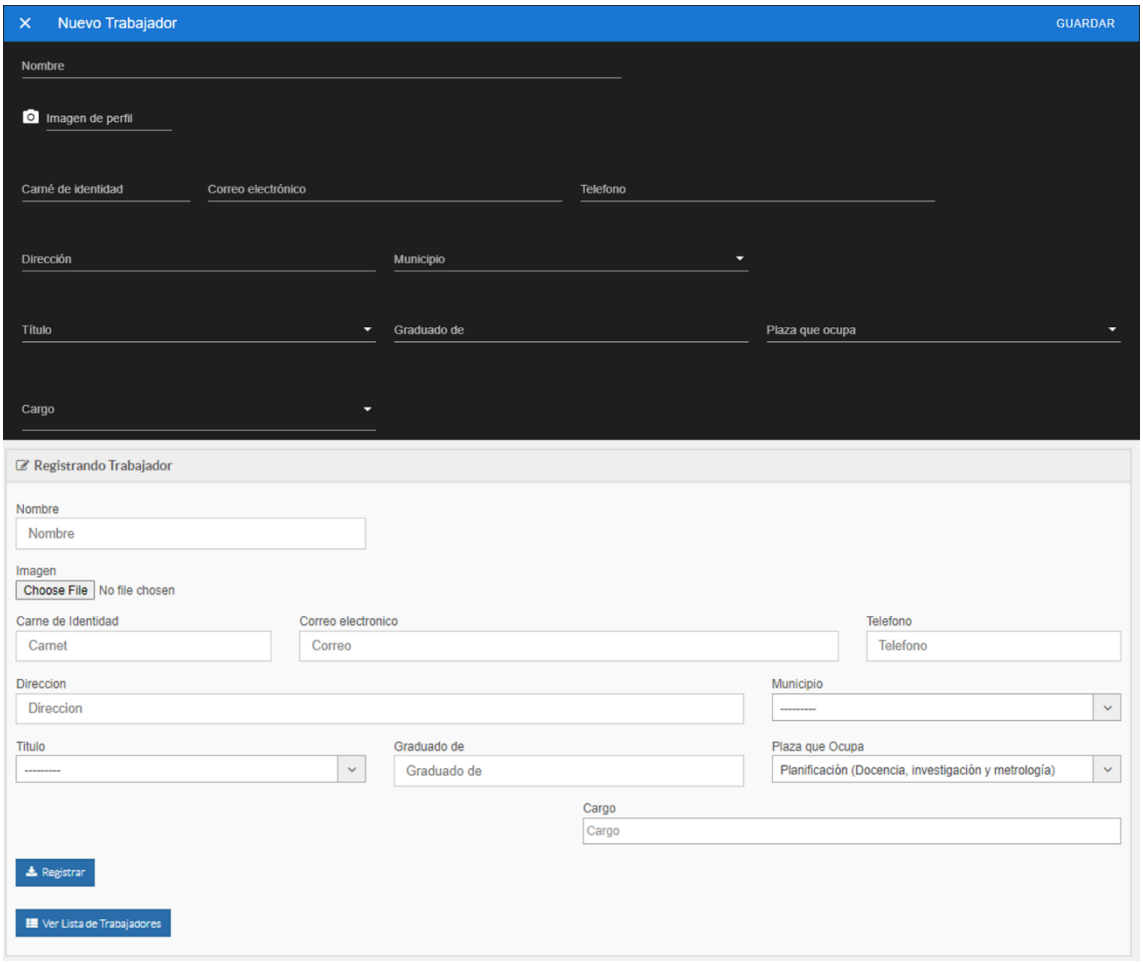


Ilustración 83 Interfaz formulario trabajador sub sistema RRHH

Información Trabajador. Antiguo/nuevo

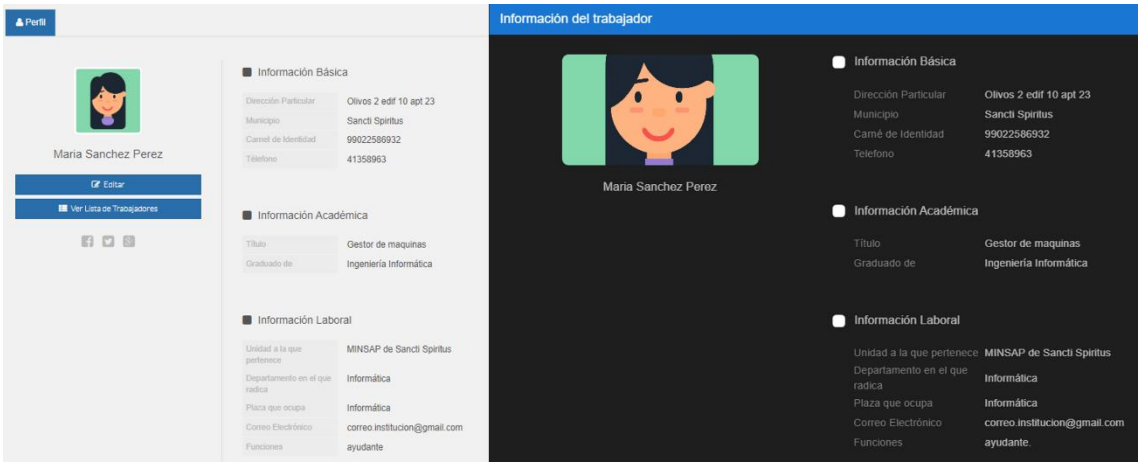
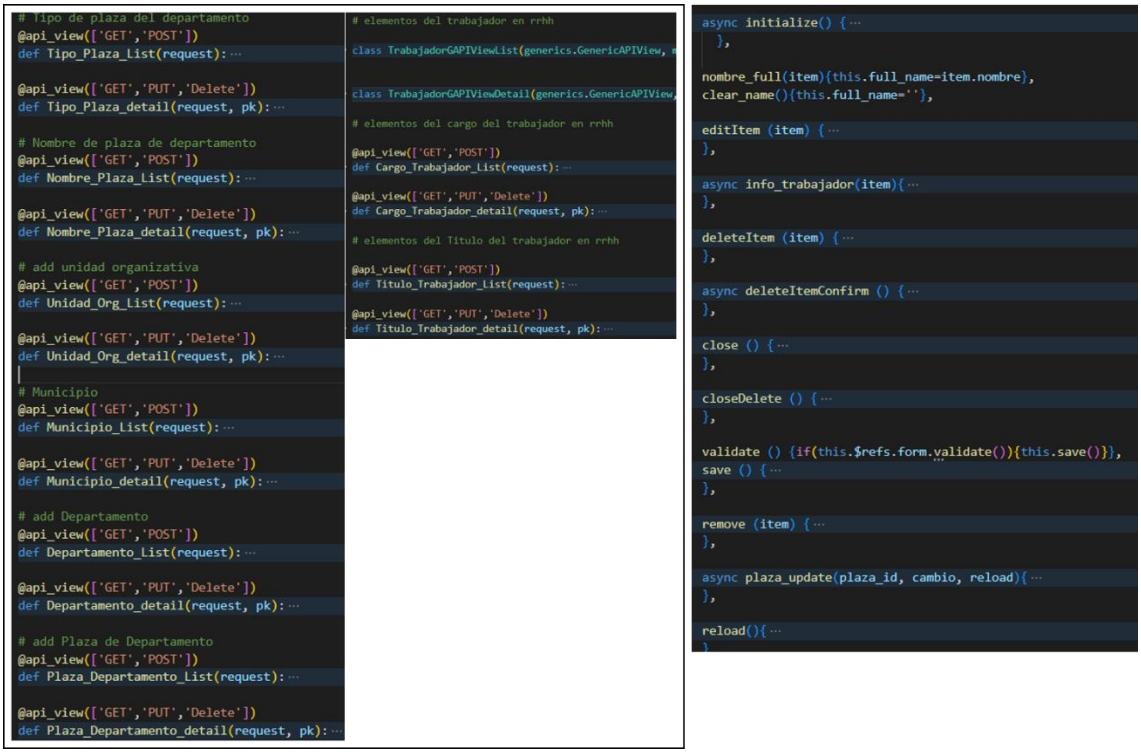


Ilustración 84 Interfaz información trabajador sub sistema RRHH

Comunicación con la interfaz

Backend/Frontend



```
# Tipo de plaza del departamento
@api_view(['GET','POST'])
def Tipo_Plaza_List(request):···

@api_view(['GET','PUT','Delete'])
def Tipo_Plaza_detail(request, pk):···

# Nombre de plaza de departamento
@api_view(['GET','POST'])
def Nombre_Plaza_List(request):···

@api_view(['GET','PUT','Delete'])
def Nombre_Plaza_detail(request, pk):···

# add unidad organizativa
@api_view(['GET','POST'])
def Unidad_Org_List(request):···

@api_view(['GET','PUT','Delete'])
def Unidad_Org_detail(request, pk):···

# Municipio
@api_view(['GET','POST'])
def Municipio_List(request):···

@api_view(['GET','PUT','Delete'])
def Municipio_detail(request, pk):···

# add Departamento
@api_view(['GET','POST'])
def Departamento_List(request):···

@api_view(['GET','PUT','Delete'])
def Departamento_detail(request, pk):···

# add Plaza de Departamento
@api_view(['GET','POST'])
def Plaza_Departamento_List(request):···

@api_view(['GET','PUT','Delete'])
def Plaza_Departamento_detail(request, pk):···
```

```
# elementos del trabajador en rrhh

class TrabajadorGAPIViewList(generics.GenericAPIView,

class TrabajadorGAPIViewDetail(generics.GenericAPIView

# elementos del cargo del trabajador en rrhh

@api_view(['GET','POST'])
def Cargo_Trabajador_List(request):···

@api_view(['GET','PUT','Delete'])
def Cargo_Trabajador_detail(request, pk):···

# elementos del Titulo del trabajador en rrhh

@api_view(['GET','POST'])
def Titulo_Trabajador_List(request):···

@api_view(['GET','PUT','Delete'])
def Titulo_Trabajador_detail(request, pk):···
```

```
async initialize() {···
  },

nombre_full(item){this.full_name=item.nombre},
clear_name(){this.full_name=''},

editItem (item) {···
},

async info_trabajador(item){···
},

deleteItem (item) {···
},

async deleteItemConfirm () {···
},

close () {···
},

closeDelete () {···
},

validate () {if(this.$refs.form.validate()){this.save()}},
save () {···
},

remove (item) {···
},

async plaza_update(plaza_id, cambio, reload){···
},

reload(){···
```

Ilustración 85 backend/frontend sub sistema RRHH

Comportamiento del sub sistema con BDD

```
# sub sistema RRHH
Feature: RRHH
    Scenario: formulario Trabajador
    Given Navegador en la pantalla de Trabajadores del Sistema integral Informatico asumiendo que este autenticado
    When Cuando el usario  Toque la función añadir Trabajador o editar Trabajador
    Then Mostrar formulario Trabajador

    Scenario: añadir Trabajador
    Given Navegador en la pantalla de Trabajadores con el formulario Trabajador desplegado del Sistema integral
    Informatico asumiendo que este autenticado
    When Cuando el usario  Toque la función Guardar
    Then Añadir Trabajador al sistema
    And Cerrar formulario Trabajador

    Scenario: editar Trabajador
    Given Navegador en la pantalla de Trabajadores con el formulario Trabajador desplegado del Sistema integral
    Informatico asumiendo que este autenticado
    When Cuando el usario  Toque la función Guardar
    Then actualizar Trabajador seleccionado al sistema
    And Cerrar formulario Trabajador

    Scenario: eliminar Trabajador
    Given Navegador en la pantalla de Trabajadores del Sistema integral Informatico asumiendo que este autenticado
    When Cuando el usario  Toque la función eliminar Trabajador
    Then Mostrar Dialogo de confirmar eliminar Trabajador
    And con la confirmación proceder a eliminar el Trabajador del sistema

    Scenario: información Trabajador
    Given Navegador en la pantalla de Trabajadores del Sistema integral Informatico asumiendo que este autenticado
    When Cuando el usario  Toque la función información del Trabajador
    Then Mostrar Dialogo con los datos del Trabajador
```

Ilustración 86 Comportamiento del sub sistema RRHH con BDD

Fase 3

Pruebas de funcionalidad (aceptación del usuario)

Tomado Bandeja de RRHH como muestra para el Test plan pruebas

Realizado Por: Ing. Julio Companioni Martínez

Ambiente de revisión: TEST

Detalles de la aplicación:

Nombre: Sistema Integral Informático

Plataforma: Django 4.0 + Nuxt.js

Base de datos: MYSQL

Desarrollado Por: Franco Salgado

Opciones de revisión

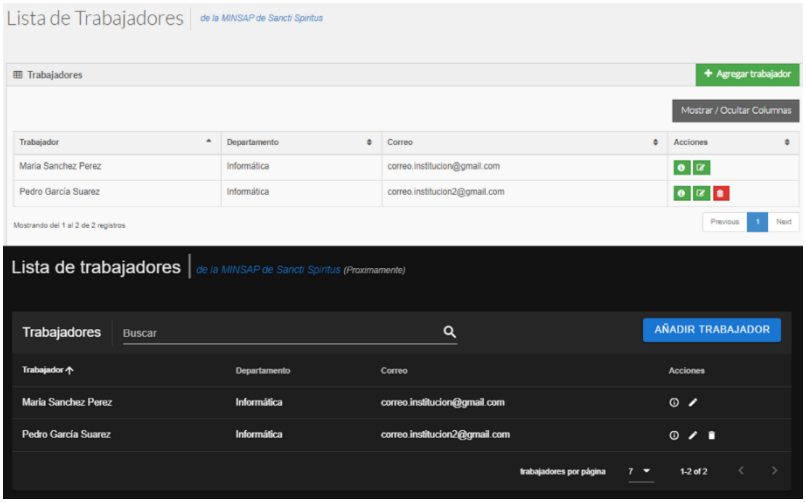


Ilustración 87 Interfaz bandeja de trabajadores sub sistema RRHH (TEST)

Opción Bandeja de trabajadores

Insertar	**Correcto**
Actualizar	Correcto
Eliminar	Correcto

Tabla 19 Opción bandeja de trabajadores RRHH(TEST)

Visualización

Las barras de desplazamientos de los campos desplegables ponerle el tema correspondiente del sistema.

Validación

Agregar dirección y correo como campos únicos.

Funcionabilidad

Cuando no haya plazas disponibles desactivar la opción de añadir trabajador.

Pruebas de rendimiento (aceptación técnica)

Añadir Trabajador: se procede a añadir un trabajador

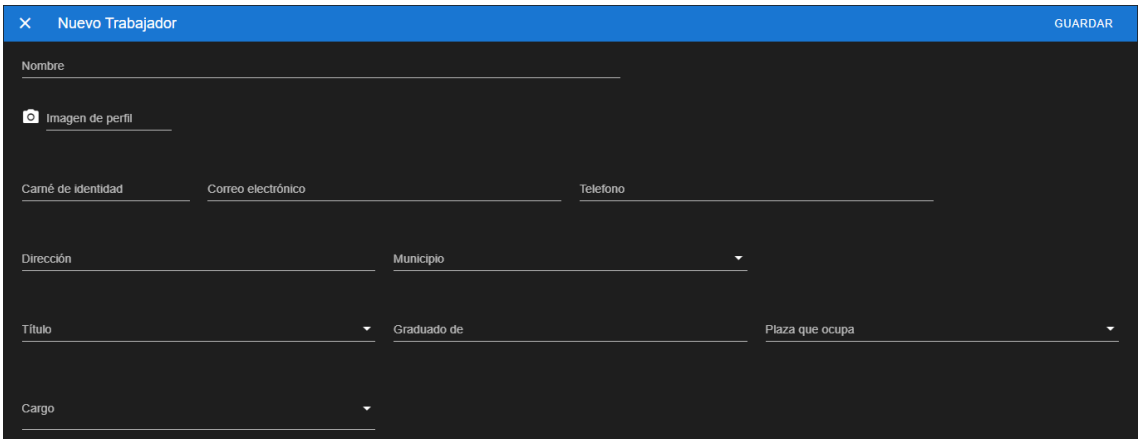


Ilustración 88 Interfaz añadir trabajador sub sistema RRHH (TEST)

Tiempo de respuesta en 489ms.

Ilustración 89 tiempo de respuesta añadir trabajador (TEST)

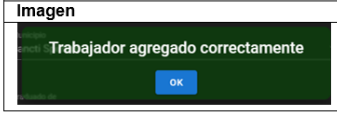
FORMATO PLAN DE PRUEBAS DE USUARIO			
Identificador	PSR01	**Versión**	V01
Responsable	Ing. Julio Companioni Martínez		
Nombre del caso de prueba	Añadir trabajador		
Módulo		**Sub módulo**	
RRHH			
Formulario			
/rrhh			
Descripción de la prueba			
Rellenar formulario Trabajador Guardar			
Resultados esperados			
Trabajador añadido			
Resultados reales			
Trabajador agregado correctamente			
Error			
Imagen			
			

Tabla 20 Prueba PSR01

Fase 4

Se precede a implementar el sub sistema terminado en un ambiente de producción. Como actualmente el Sistema Integral Informático no está prestando servicio no es una complicación ponerlo en producción. En las fases anteriores quedó definido su modelo, su esquema BDD y se procede ahora a realizar la migración de la base de datos y el sub sistema.

Anexo 7: Manual de usuario

Proceso de inicialización previa antes de poner en consumo el sistema

Primeramente, se debe crear al súper usuario que tendrá acceso al panel administrativo de django de la siguiente forma:

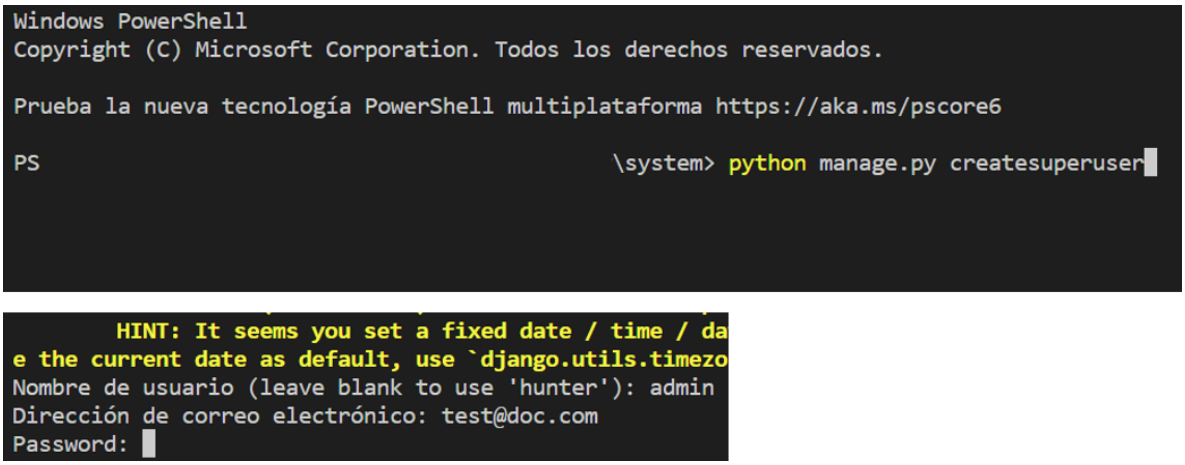


Ilustración 90 Creación del súper usuario

Una vez creado el súper usuario se procede a acceder al panel administrativo de django con las credenciales recién creadas. En este punto se debe proceder a dejan los elementos predefinidos que tiene que tener el sistema

- Categorías de mensajes: deben estar en el sistema las categorías con las que contarán los mensajes que pasan a través del sistema.

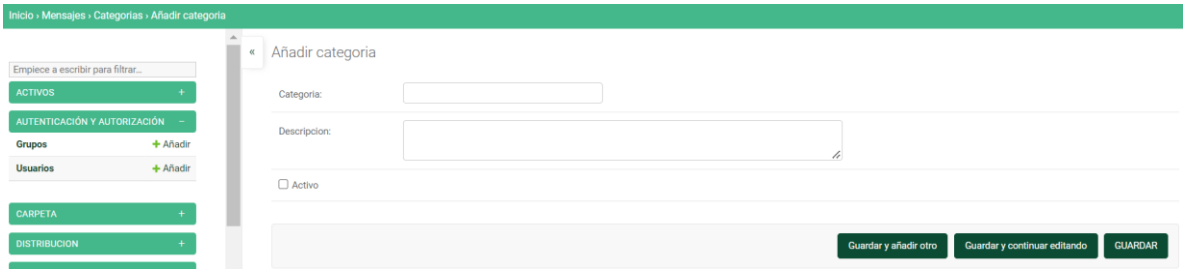


Ilustración 91 Django admin: Categorías de mensajes

- Provincias: Añadir las provincias del país.

Ilustración 92 Django admin: Provincias

- Municipios: Añadir los municipios que integran las provincias.

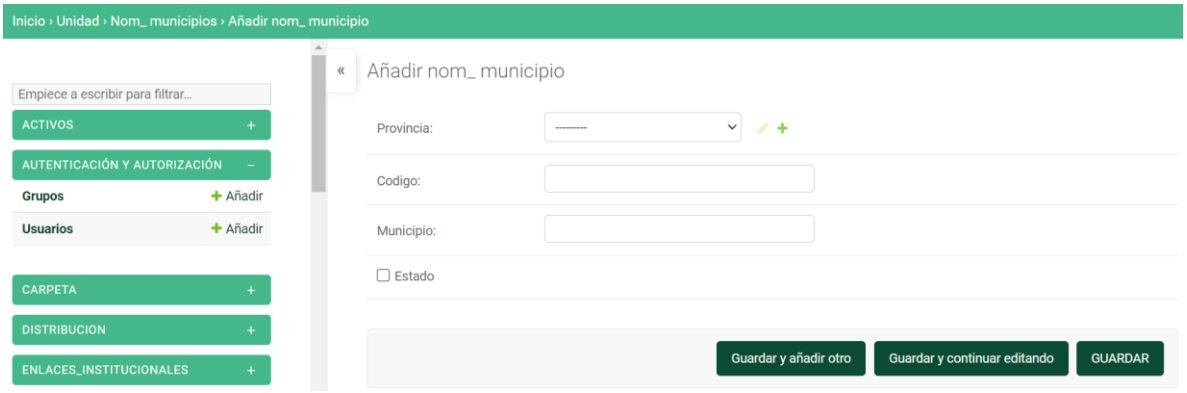


Ilustración 93 Django admin: municipios

- Tipos de unidades: Establecer todos los tipos de unidades que puede tener el sistema

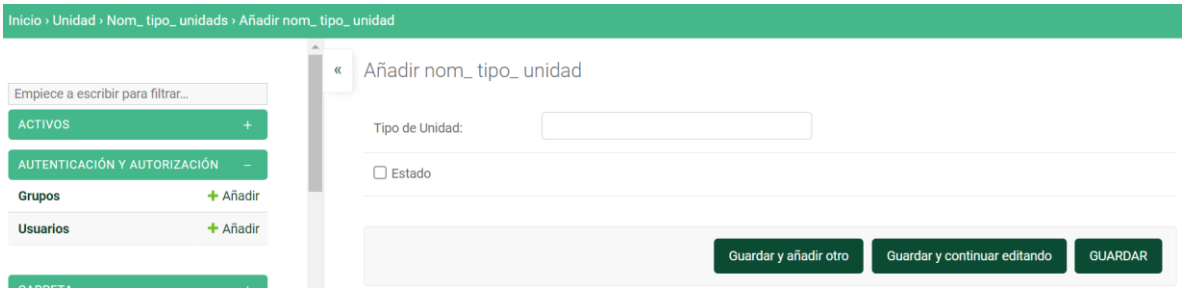


Ilustración 94 Django admin: tipos de unidades

- Perfiles de unidades: Establecer los diferentes perfiles que pueden tener las unidades

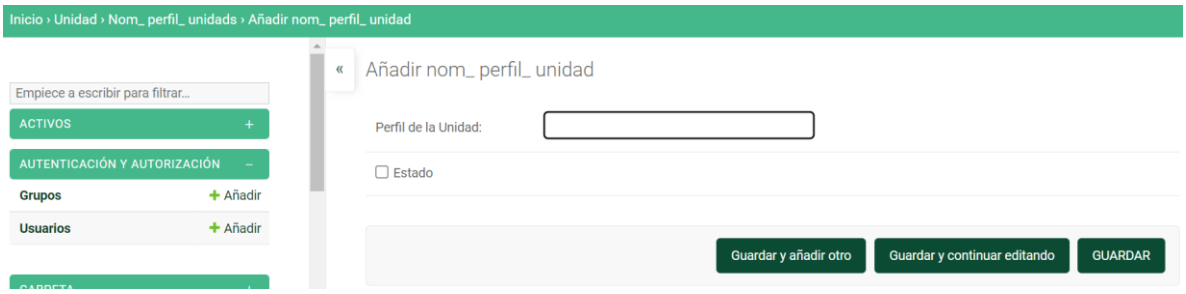


Ilustración 95 Django admin: perfiles de unidades

Con los datos anteriores se puede pasar ya para establecer las unidades que tiene el sistema. Estas unidades se crean mediante el administrador de django exclusivamente.

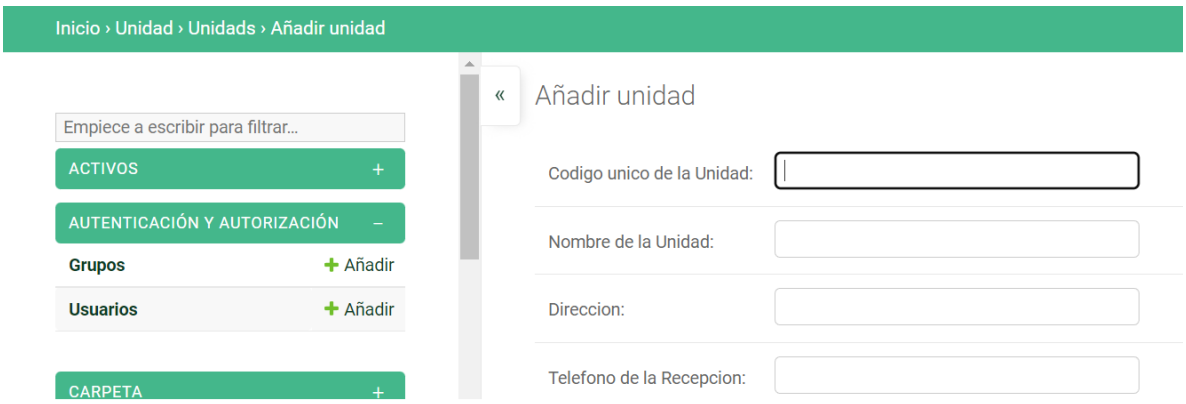


Ilustración 96 Django admin: unidades

El próximo paso es establecer los valores de los elementos relacionados con los departamentos. Para ello deben quedar establecidos:

- Los nombres de las plazas: Incluir todos los nombres que pueden tomar las plazas de los departamentos.

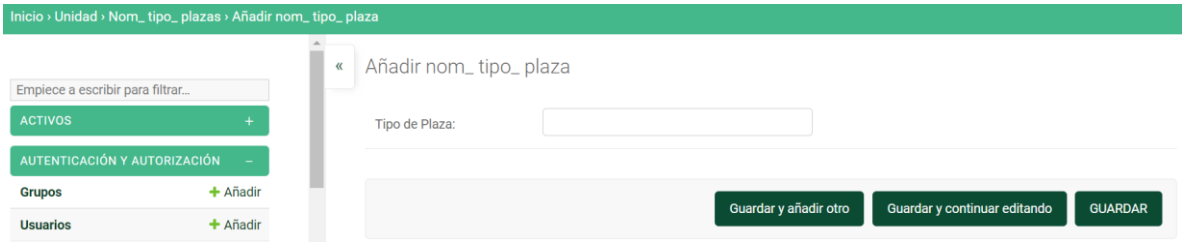


Ilustración 97 Django admin: Nombres de plazas

- Los datos de las plazas de los departamentos:

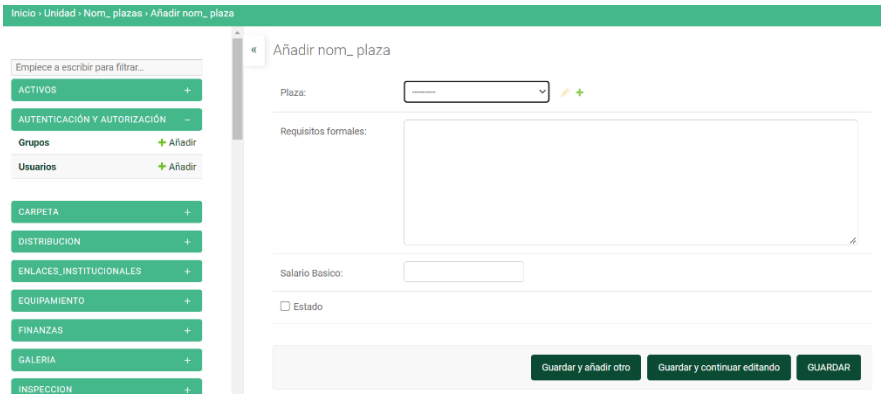


Ilustración 98 Django admin: Datos de plazas

- Todos los tipos de unidades organizativas:

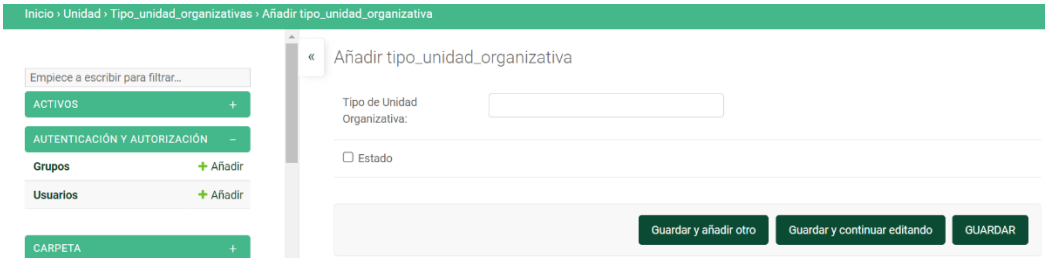


Ilustración 99 Django admin: tipos de unidades organizativas

- Todas las unidades organizativas de la empresa. Estas unidades organizativas decidirán el rol que va a tener el usuario dentro del sistema, actualmente están definidas:
    - Informática
    - Planificación
    - Transporte
    - Contabilidad
    - Higiene
    - Mantenimiento y Sistemas de Ingeniería
    - Docencia, Investigación y metrología

Estas pueden ser reemplazadas fácilmente en el código.

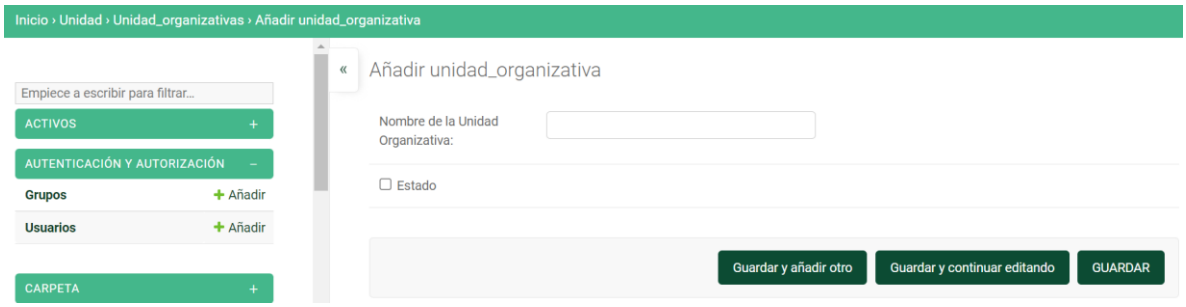


Ilustración 100 Django admin: unidades organizativas

En este punto se procede a añadir los departamentos que componen al sistema.

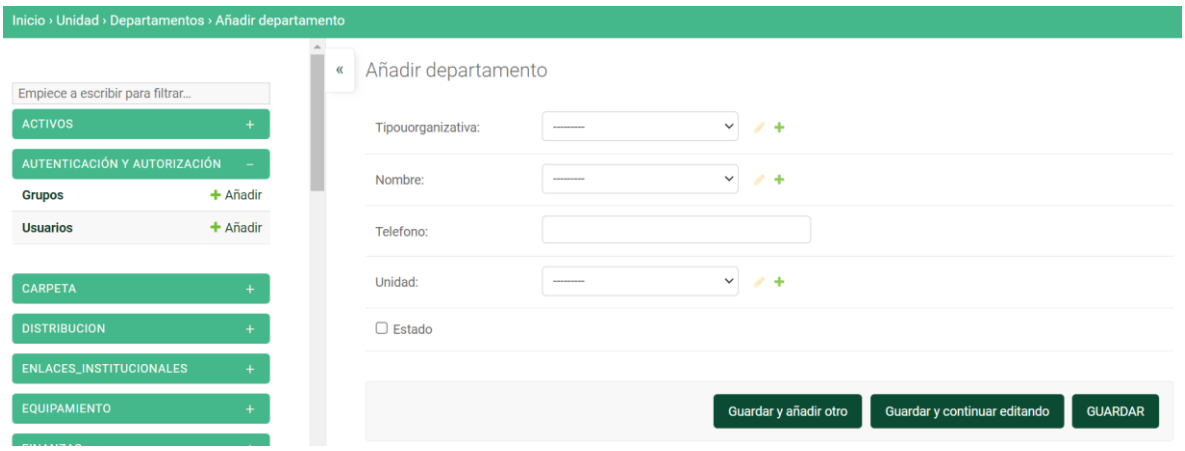


Ilustración 101 Django admin: Departamentos

Se procede a crear la primera y única plaza de departamento, como más adelante se creará el primer trabajador mediante django admin esta plaza debe tener como estado “ocupada” y debe ser del departamento que de nombre sea “informática” pues el primer trabajador será de la rama de informática.

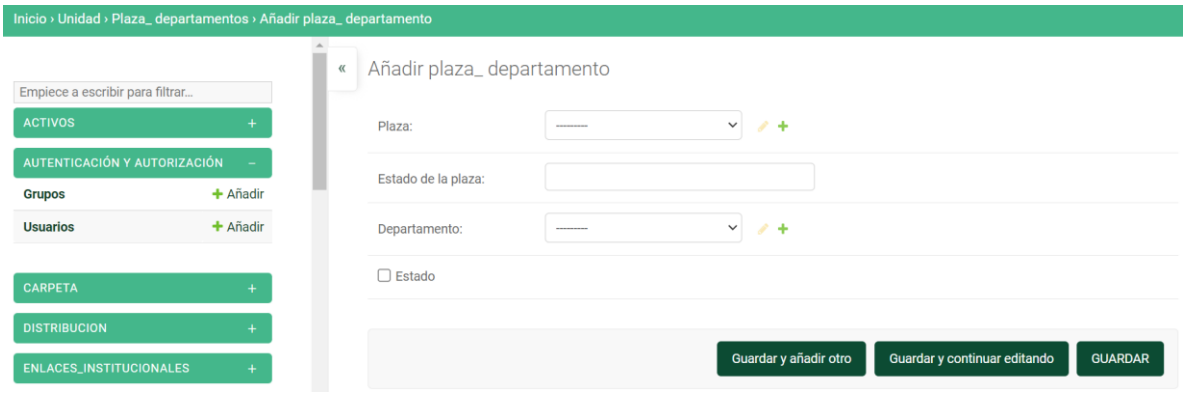


Ilustración 102 Django admin: Plazas de departamentos

Ya quedó la parte de departamentos concluida y se procede a la preparación del primer trabajador en el sistema para ello como requisitos previos debe estar definidos:

- Todos los títulos que pueden presentar los trabajadores.

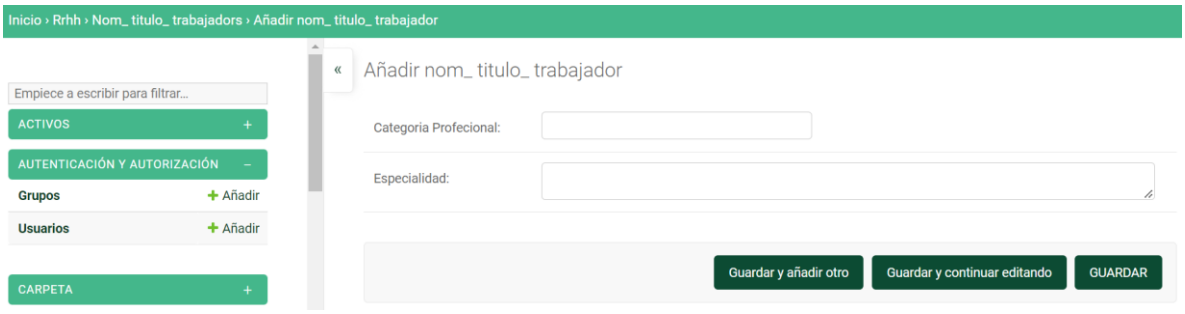


Ilustración 103 Django admin: Títulos de trabajadores

- Todos los cargos que pueden presentar los trabajadores.

Ilustración 104 Django admin: Cargos de trabajadores

- Usuario: Se crea el primer usuario del sistema para este primer trabajador. El sistema de usuarios de este sistema se gestiona mediante django admin, es independiente a los trabajadores, no todos los trabajadores tendrán acceso a un usuario y cuando se cree un trabajador que requiera un usuario se debe ir con el administrador del sistema

a que le proporcione un usuario.

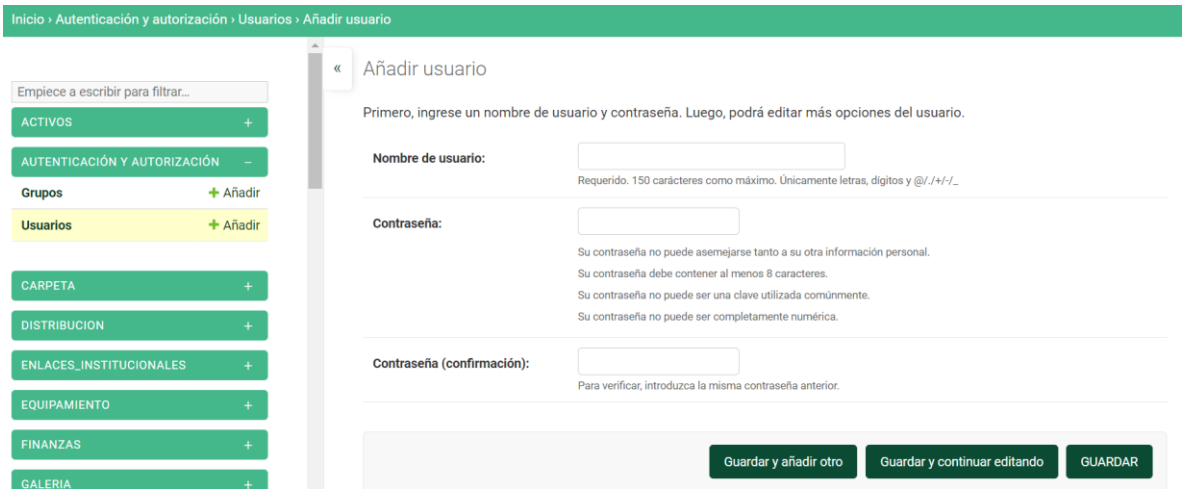


Ilustración 105 Django admin: Usuarios

Ya con los datos anteriores se está listo para crear al primer trabajador de la entidad. El informático, que se encargará de las gestiones de añadir trabajadores, plazas, departamentos en el sistema.

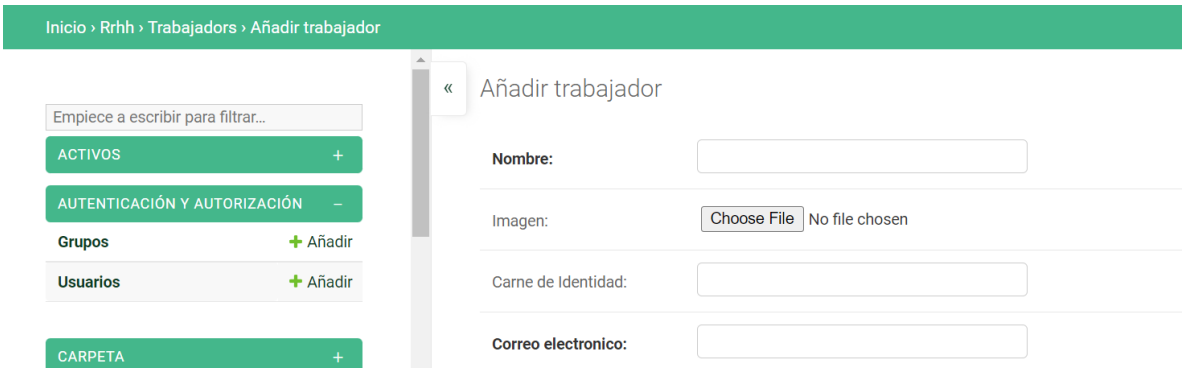


Ilustración 106 Django admin: Trabajadores

Así concluye la parte de inicialización del Sistema Integral Informático.

Manejo de la sección de mensajes

Objetivo: comunicación entre los usuarios del sistema.

Composición de la pantalla: Se divide en cuatro bloques.

Buzón de entrada: Espacio donde se muestran los mensajes de entrada.

- Icono de cesto de basura: enviar mensaje a bandeja de eliminados

- Abrir: Mostrar mensaje seleccionado con sus datos.

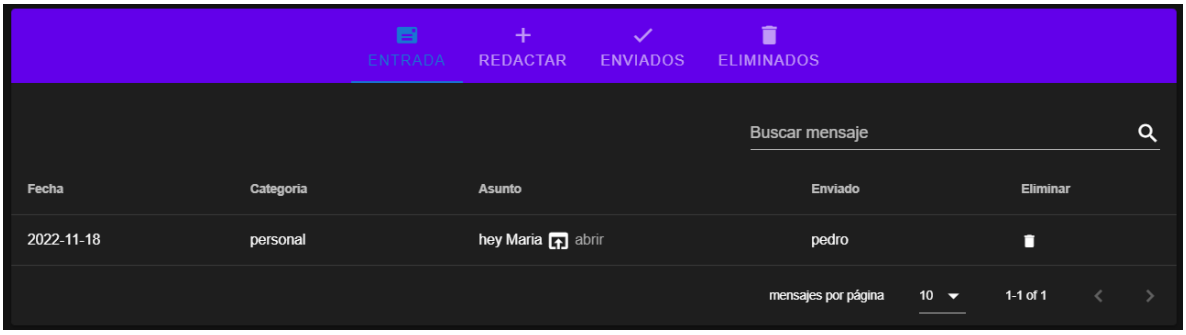


Ilustración 107 Manual de usuario: entrada Mensajes

Buzón de enviados: Espacio donde se muestran los mensajes enviados.

- Icono de cesto de basura: enviar mensaje a bandeja de eliminados
- Abrir: Mostrar mensaje seleccionado con sus datos.

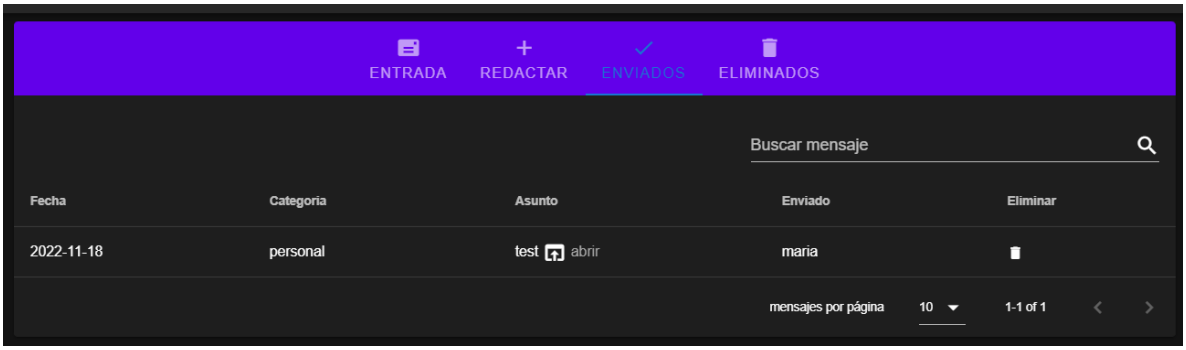


Ilustración 108 Manual de usuario: enviados Mensajes

Buzón de eliminados: Espacio donde se muestran los mensajes eliminados.

- Icono de reciclaje: restaura el mensaje a su origen.
- Icono de cesto de basura: elimina permanentemente el mensaje.

- Abrir: Mostrar mensaje seleccionado con sus datos.

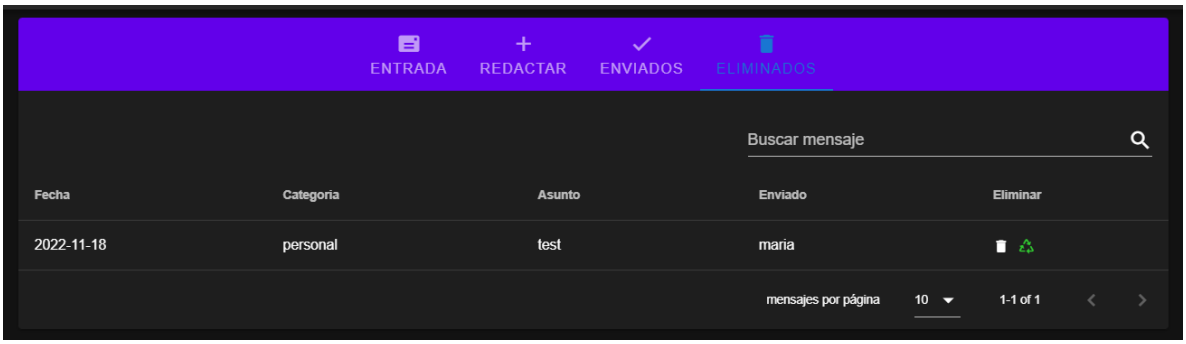


Ilustración 109 Manual de usuario: eliminados Mensajes

Redactar: Enviar mensaje a uno o varios destinatarios.

- Categoría: Se selecciona la categoría que tendrá el mensaje
- Enviar: Se procede a enviar el mensaje a los respectivos destinatarios
- Cancelar: Se limpia los campos para emitir un nuevo mensaje.

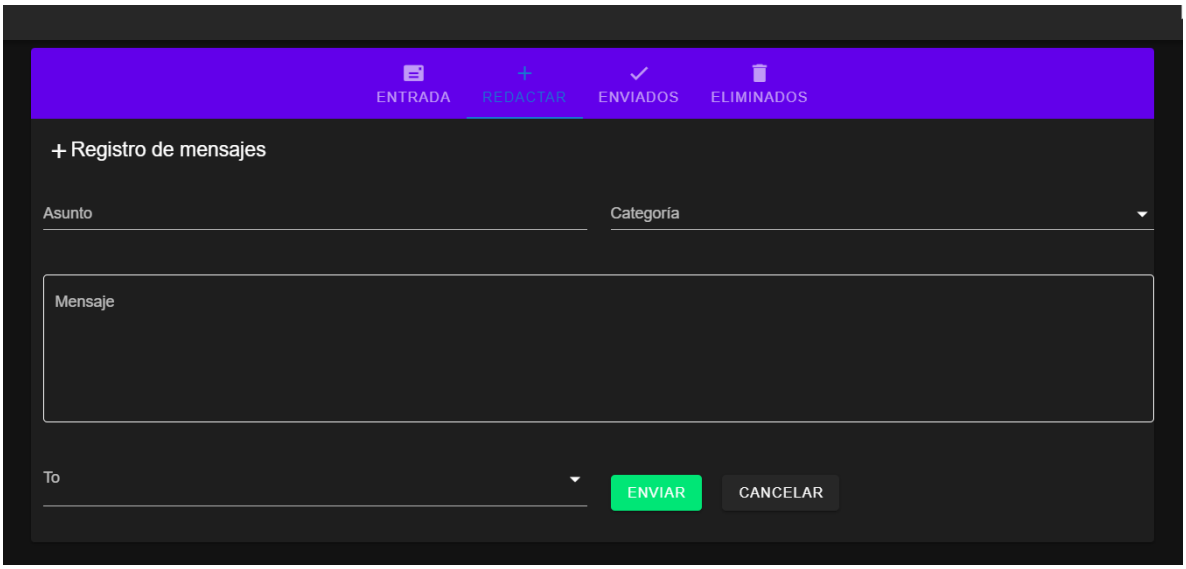


Ilustración 110 Manual de usuario: redactar Mensajes

Manejo de la sección de Enlaces Institucionales.

Objetivo: administración de Enlaces Institucionales.

Composición de la pantalla: Se divide en dos bloques.

Bandeja de Enlaces Institucionales: Espacio donde se muestran los Enlaces Institucionales.

- Icono de Lápiz: Editar datos del enlace seleccionado
- Icono de cesto de basura: Eliminar enlace seleccionado
- Abrir: Mostrar observaciones que posee el enlace

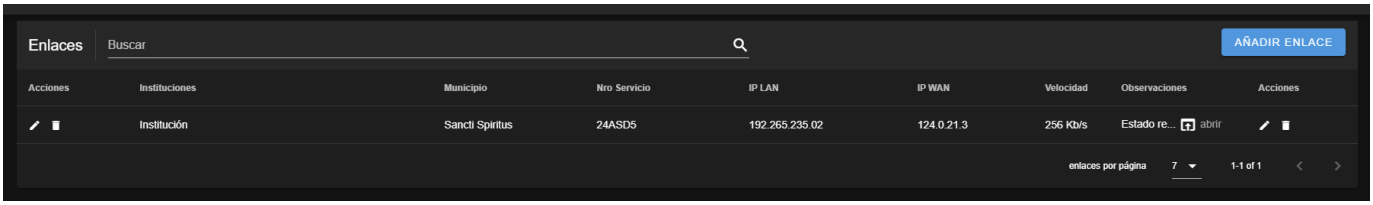


Ilustración 111 Manual de usuario: entrada Enlaces

Añadir/editar Enlace Institucional: Espacio para agregar o modificar un enlace institucional.

- Observaciones: añadir una observación al enlace en cuestión.
- Cancelar: Se cancela la creación/edición del enlace actual.
- Guardar: Se almacena/modifica el enlace en cuestión en el sistema.

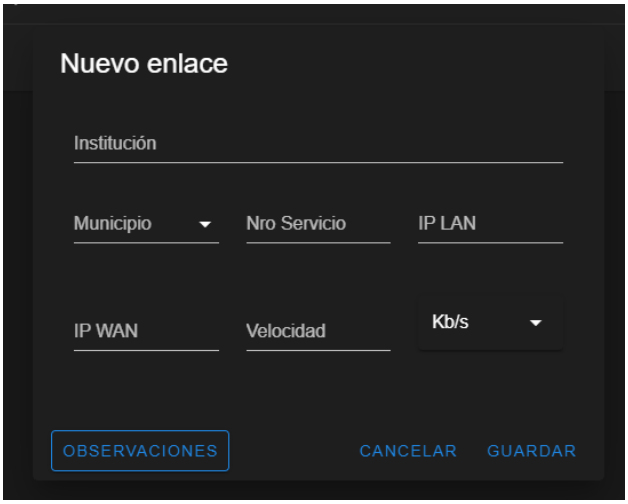


Ilustración 112 Manual de usuario: añadir/editar Enlaces

Manejo de la sección de Departamentos.

Objetivo: administración de Departamentos.

Composición de la pantalla: Se divide en dos bloques.

Bandeja de Departamentos: Espacio donde se muestran los Departamentos.

- Icono de Lápiz: Editar datos del departamento seleccionado.
- Icono de cesto de basura: Eliminar departamento seleccionado.
- Icono de información: Mostrar información del departamento seleccionado.

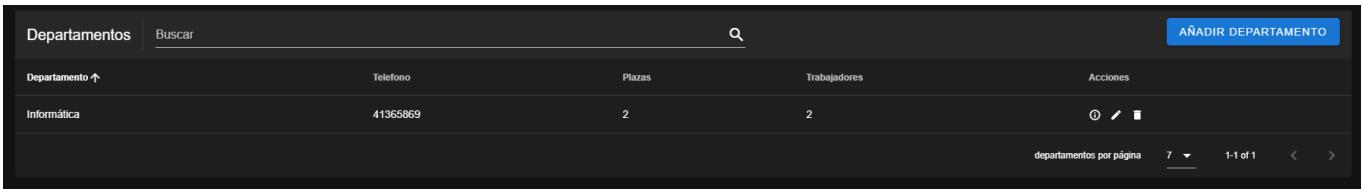


Ilustración 113 Manual de usuario: entrada Departamentos

Añadir/editar Departamento: Espacio para agregar o modificar un departamento.

- Cancelar: Se cancela la creación/edición del departamento actual.
- Guardar: Se almacena/modifica el departamento en cuestión en el sistema.

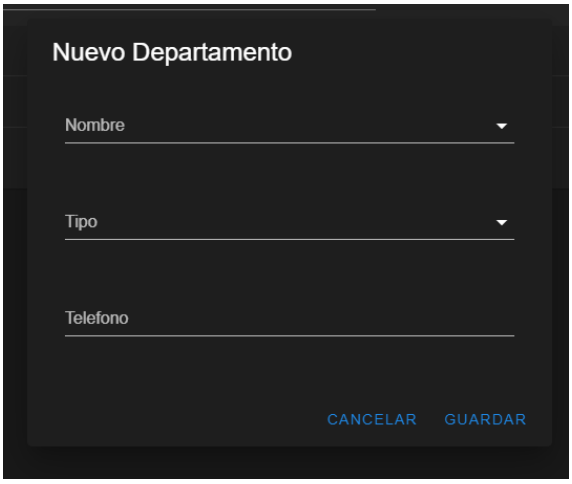


Ilustración 114 Manual de usuario: añadir/editar Departamentos

Manejo de la sección de Plazas.

Objetivo: administración de Plazas.

Composición de la pantalla: Se divide en dos bloques.

Bandeja de Plazas: Espacio donde se muestran las Plazas.

- Icono de Lápiz: Editar datos de la plaza seleccionada.
- Icono de cesto de basura: Eliminar plaza seleccionada.

- Icono de información: Mostrar información de la plaza seleccionada.

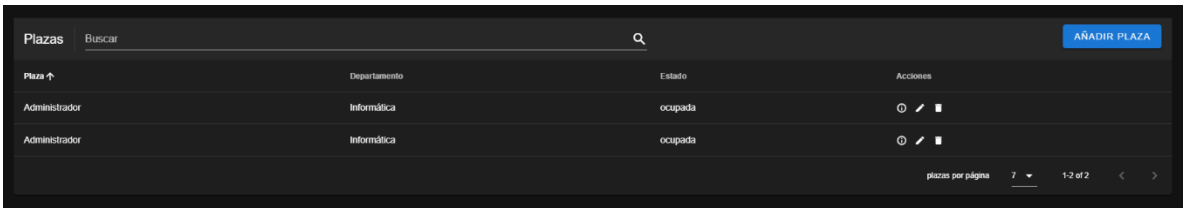


Ilustración 115 Manual de usuario: entrada Plazas

Añadir/editar Plaza: Espacio para agregar o modificar una plaza.

- Cancelar: Se cancela la creación/edición de la plaza actual.
- Guardar: Se almacena/modifica la plaza en cuestión en el sistema.

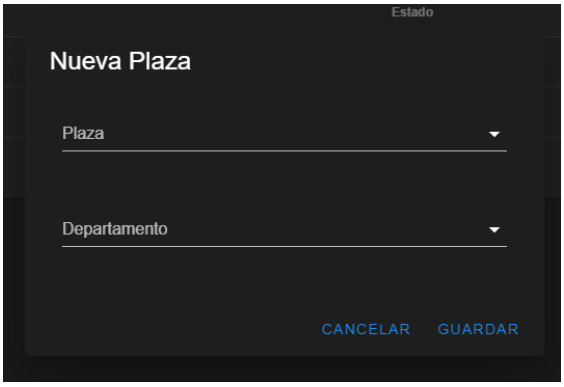


Ilustración 116 Manual de usuario: añadir/editar Plazas

Manejo de la sección de RRHH.

Objetivo: administración de RRHH.

Composición de la pantalla: Se divide en dos bloques.

Bandeja de RRHH: Espacio donde se muestran los Trabajadores.

- Icono de Lápiz: Editar datos del trabajador seleccionado.
- Icono de cesto de basura: Eliminar trabajador seleccionado.
- Icono de información: Mostrar información del trabajador seleccionado.

- Si no hay plazas disponibles no se podrá añadir un nuevo trabajador.

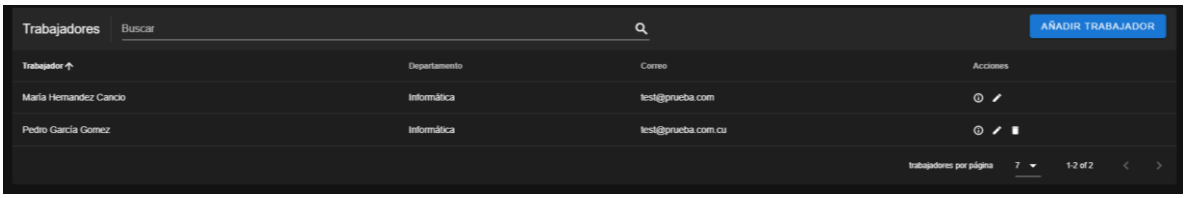


Ilustración 117 Manual de usuario: entrada RRHH

Añadir/editar Trabajador: Espacio para agregar o modificar un trabajador.

- Cancelar: Se cancela la creación/edición del trabajador actual.
- Guardar: Se almacena/modifica el trabajador en cuestión en el sistema.

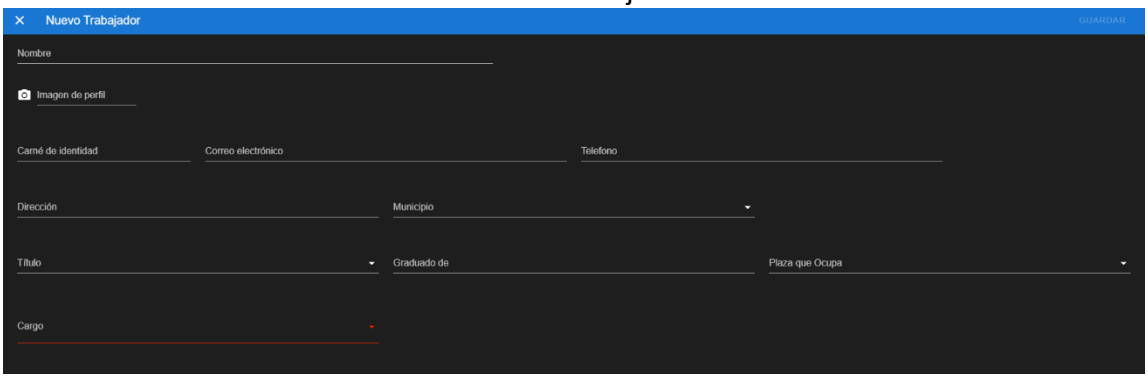


Ilustración 118 Manual de usuario: añadir/editar RRHH