ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

DISEÑO E IMPLEMENTACIÓN DE UNA ARQUITECTURA CORE PARA AUTOMATIZACIÓN Y MEJORA DE SERVICIOS EN LOS LABORATORIOS DE LA FACULTAD DE INGENIERÍA EN SISTEMAS UTILIZANDO LA METODOLOGÍA SCRUM

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SOFTWARE

Alex Javier Erazo Barrera

alex.erazo01@epn.edu.ec

Anderson Fernando Cárdenas Real

anderson.cardenas@epn.edu.ec

DIRECTOR: Ing. Rodrigo Fabián Chancusig Chuquilla, Msc.

rodrigo.chancusig@epn.edu.ec

DMQ, agosto 2022

CERTIFICACIONES

Nosotros, Alex Javier Erazo Barrera y Anderson Fernando Cárdenas Real declaramos que el trabajo de integración curricular aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Firmado electrónicamente por:
ALEX JAVIER
ERAZO

Alex Javier Erazo Barrera

Firmado electrónicamente por:
ANDERSON
FERNANDO
CARDENAS REAL

Anderson Fernando Cárdenas Real

Certifico que el presente trabajo de integración curricular fue desarrollado por Alex Javier Erazo Barrera y Anderson Fernando Cárdenas Real, bajo mi supervisión.

Firmado electrónicamente por:
RODRIGO FABIAN
CHANCUSIG
CHUQUILLA

Ing. Rodrigo Fabián Chancusig Chuquilla, MSc.

DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Alex Javier Erazo Barrera

Anderson Fernando Cárdenas Real

Ing. Rodrigo Fabián Chancusig Chuquilla, MSc.

DEDICATORIA

Dedico todo mi empeño y esfuerzo a la persona que siempre creyó en mí y lo sigue haciendo sin dudar, a mi amada madre Luz María Barrera.

Alex Javier Erazo Barrera

…para aquel niño que algún día soñó con saber algo sobre el mundo,

para la pequeña que me mira como ejemplo y siempre me espera

y finalmente para la señorita del otro lado de la ciudad.

Anderson Fernando Cárdenas Real

AGRADECIMIENTO

Agradezco a todas las personas que fueron parte de este duro camino, en especial a mi hermano Henry Erazo y a mi padrino Jorge Lugmaña, quienes siempre supieron darme buenos consejos y su apoyo. A mis mejores amigos José Latorre y Rony Pantoja que siempre creyeron en mí y me apoyaron desde pequeño. Por último, a la Corporación Favorita y ex compañeros de trabajo, quienes me dieron la apertura para poder continuar y finalmente terminar mis estudios profesionales en una etapa muy importante de mi vida.

Alex Javier Erazo Barrera

Esta sección es para mi madre que siempre me apoyó sin importar las circunstancias, siempre miraste por mi bienestar y me cuidaste en el camino, un día me retirabas de la escuela, al siguiente día me llevabas por primera vez al colegio y hoy solo me queda agradecer a Dios por tenerte en mi vida. Sin ti esto no habría sido posible, que la vida me alcance para devolverte todo el amor.

Anderson Fernando Cárdenas Real

ÍNDICE DE CONTENIDO

RESUMEN

En el presente documento se describe el diseño e implementación de una arquitectura CORE, la cual será útil para los futuros desarrollos de sistemas que apoyen en los procesos de los laboratorios de la Facultad de Ingeniería en Sistemas (LAB-FIS) de la Escuela Politécnica Nacional (EPN) y el desarrollo de una interfaz de programación de aplicaciones (API) REST que permite el acceso de datos. Se abarcaron cuatro módulos de datos: inventario, préstamo de equipos, préstamo de servicios virtuales y asistencias técnicas. Además, se implementó un sistema de autenticación y autorización basado en roles.

Para todo el proceso se utilizó la metodología de desarrollo SCRUM en conjunto con técnicas de desarrollo ágil como user story mapping. En el apartado de las pruebas se realizaron pruebas unitarias a la API. Finalmente, se realizó el despliegue en los servidores de LAB-FIS y se ejecutaron pruebas de estrés sobre la infraestructura para medir el rendimiento.

PALABRAS CLAVE: Ingeniería de software, arquitectura core, desarrollo de software.

ABSTRACT

This document describes design and implementation of a CORE architecture, which will be useful for future development of systems that support processes of the Faculty of Systems Engineering laboratories (LAB-FIS) of the National Polytechnic School (EPN) and the development of a REST application programming interface (API) that allows data access. Four data modules were covered: inventory, equipment loan, virtual services loan, and technical assistance. In addition, a role-based authentication and authorization system was implemented.

SCRUM development methodology was used for the entire process in conjunction with agile development techniques such as user story mapping. In the testing section, unit tests were performed on the API. Finally, the deployment was performed on LAB-FIS servers and stress tests were executed on the infrastructure to measure performance.

KEYWORDS: Software engineering, Core arquitecture, software development.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

En la actualidad las organizaciones han hecho uso de sistemas informáticos para la automatización de procesos y gestión de servicios, esto les ha permitido innovarse y potenciar sus capacidades. Además, los equipos computacionales han llegado a ser una herramienta indispensable que, en conjunto con una plataforma de software apropiada, optimizan y controlan los recursos utilizados en las diferentes actividades, procesos o servicios. [1]

Por otro lado, el manejo adecuado de los datos de una entidad es crucial en muchos aspectos; por ejemplo, para la toma de decisiones, control, monitoreo o evaluación. Por este motivo se debe considerar siempre los parámetros de seguridad, tanto en la aplicación de software, la infraestructura y sobre todo en los datos. [2]

Por lo antes dicho, nace la necesidad de centralizar las diferentes fuentes de datos, considerando las medidas de seguridad adecuadas para su acceso y almacenamiento. Una arquitectura CORE contempla dichos aspectos, siendo capaz de gestionar todos los registros en un mismo punto de acceso, a partir del cual se puede implementar sistemas software corporativos. [3]

Para todo desarrollo de software, incluyendo la implementación de una arquitectura CORE, es necesario contar con una metodología adecuada. SCRUM conserva un enfoque ágil que involucra al cliente e interesados en el proceso. [4]

Los Laboratorios de la Facultad de Ingeniería de Sistemas (LAB-FIS) de la Escuela Politécnica Nacional es una organización que lleva a cabo una serie de procesos con el fin de dar una asistencia técnica a profesores, estudiantes, personal externo e interno de la facultad. Actualmente no se cuenta con un sistema informático el cual pueda asistir en las actividades que lleva a cabo, esto permitiría ser más eficiente en sus procesos proporcionando información adecuada y oportuna tanto para la planificación como para la toma de decisiones.

Con el fin de solventar esta problemática, se realizó la implementación de una arquitectura CORE que permitirá la futura automatización de los procesos de inventario, reserva de equipos, gestión de solicitudes de asistencia técnica y solicitudes de servicios virtuales. El componente será capaz de ofrecer servicios que permitan el acceso, registro y actualización de los datos necesarios para llevar a cabo los procesos antes mencionados con las medidas de seguridad adecuadas.

Finalmente, para la elaboración de lo propuesto se utilizará un método de investigación con dos enfoques, cuantitativamente se centrará en el rendimiento del producto y la eficacia de su funcionalidad, tomando métricas adecuadas que permitan medir y respaldar la calidad del producto. En cuanto a lo cualitativo se tomará en cuenta la conformidad del personal respecto a los resultados obtenidos, con la finalidad de solventar los problemas presentes en los procesos y servicios.

1.1 Objetivo general

Diseñar e implementar una arquitectura CORE para LAB-FIS que permita el manejo eficiente de los datos para la toma de decisiones.

1.2 Objetivos específicos

Para este trabajo se ha contemplado cumplir con los siguientes objetivos específicos:

1. Analizar diferentes patrones de arquitectura para determinar un diseño adecuado y acorde a las necesidades de LAB-FIS.
2. Elicitar los requerimientos de los procesos de inventario, reserva de equipos, gestión de solicitudes de asistencias técnicas y solicitudes de servicios virtuales en conjunto con el personal de los laboratorios.
3. Modelar y desarrollar la base de datos que abarque los procesos principales de LAB-FIS y la gestión de usuarios.
4. Desarrollar una interfaz de programación de aplicaciones (API) por la cual se pueda acceder, registrar y modificar los datos necesarios para el flujo de los procesos principales de LAB-FIS.
5. Realizar el despliegue de la API y la base de datos en los servidores de LAB-FIS.
6. Verificar el funcionamiento de los componentes construidos sobre la arquitectura de LAB-FIS.

1.3 Alcance

Se contempla finalizar el proyecto con el diseño e implementación de una API que permita crear, actualizar y modificar los datos relacionados a:

- Inventario
- Servicios de préstamos de equipos
- Solicitudes de servicios virtuales
- Registros de asistencias técnicas

Luego de esto, se realizarán las configuraciones de seguridad dentro de la arquitectura para finalmente hacer el despliegue en los servidores de LAB-FIS.

1.4 Marco teórico

Exponer el marco teórico relevante relacionado con el tema, incluyendo los argumentos que justifican la validez de lo realizado, con una revisión bibliográfica pertinente.

SCRUM

SCRUM es una metodología ágil para desarrollar productos y servicios innovadores basados en los principios del agilismo, que de manera sencilla se pueden ver en el manifiesto ágil que dicta lo siguiente:

“Los individuos y su interacción, por encima de los procesos y las herramientas.

El software que funciona, por encima de la documentación exhaustiva.

La colaboración con el cliente, por encima de la negociación contractual.

La respuesta al cambio, por encima del seguimiento de un plan.”

Para lograr la creación de un producto software primero se necesita tener un Product Backlog, lo cual es una lista priorizada de las características y otras cualidades que debe tener nuestro producto final, este Product Backlog es generado a partir de las entrevistas y aproximaciones que se tiene con el cliente cuando expresa los requerimientos iniciales, estos requerimientos iniciales no deben ser tan exhaustivos y detallados pero si lo suficientemente específicos como para poder tener una idea general del producto y las

características mínimas que este debe contener. El tiempo que se estima para poder generar este Product Backlog debe estar entre 1 semana y 1 mes.

Dentro de los beneficios de la aplicación de SCRUM como metodología de desarrollo tenemos:

1. Clientes contentos debido a que obtienen avances constantes y reales del producto que desean.
2. Mejor retorno de inversión
3. Reducción de costos
4. Obtención rápida de resultados
5. Confianza de éxito a pesar de un entorno complejo
6. Mayor diversión e interacción debido a la estructura de la metodología

Esta metodología define claramente tres roles:

1. Product owner: Es el encargado de representar a todos los interesados y busca que el producto cumpla con las expectativas del cliente.
2. SCRUM máster: Es el gurú de la metodología conoce los detalles y permite que el proyecto siga la metodología para obtener el mejor producto posible.
3. Equipo de desarrollo: Equipos multidisciplinarios y autogestionados que se encargarán del diseño, implementación, prueba y puesta en marcha del producto.

Las fases que tiene la metodología son las siguientes:

1. Sprint planning y Sprint backlog: El product backlog puede ser a muy alto nivel y no tener claro cuáles son exactamente los entregables precisos que se necesitan para el cumplimiento de la fase del producto de manera tangible, entonces en esta fase se define a detalle las historias y tareas allevar a cabo, así como su priorización para poder completar a tiempo los entregables.
2. Sprint execution: Son las dos semanas en las que el equipo de desarrollo se encarga plenamente de la implementación en código de las tareas que se especifican.
3. Daily SCRUM: Es una pequeña reunión diaria que se tiene con el equipo, se la realiza de pie y se satisfacen las siguientes preguntas:

- ¿Qué hiciste ayer?
- ¿Qué harás hoy?
- ¿Qué problemas tienes?

Estas reuniones permitirán conocer de manera inmediata los problemas que los miembros del equipo tengan y resolver estos inconvenientes de forma inmediata para evitar bloqueos.

4. Sprint review: Es una fase de revisión en la que tenemos una conversación con el equipo SCRUM y demás interesados en el que ya se tiene una parte funcional, se realizan demostraciones y explicaciones, de ser necesario se explicarán los cambios o adaptaciones que el cliente solicite y estas modificaciones serán prioritarias para el siguiente Sprint. [4]

Protocolo de transferencia de hipertexto (HTTP)

El protocolo de transferencia de hipertexto (HTTP) se encuentra entre los protocolos que posee una velocidad adecuada para sistemas de información distribuidos y colaborativos. Es sin estado, orientado a objetos y genérico, por lo tanto, se puede usar para diversas situaciones con la ayuda de sus métodos de solicitud. Por otro lado, nos permite construir sistemas sin preocuparse de la transferencia de datos y altamente confiables. [5]

Servicios web

De acuerdo con [6], un servicio web se define como aquel software que es capaz de realizar la interacción entre máquinas por medio de la red. Cada servicio web tiene una funcionalidad específica o varias con las que cumple y por lo general para interactuar con un servicio es necesario conocer el formato por el cual se envió y/o retorna información, además de los protocolos que maneja.

Interfaz de programación de aplicaciones (API)

Una API permite comunicar diferentes sistemas o servicios, su principal beneficio es que ocultan la implementación y ofrecen entradas simplificadas y transparentes. Representan

la frontera entre las aplicaciones de los clientes con los sistemas de backend y utilizan un lenguaje de transporte para intercambiar información entre ambos. [7]

Bases de datos relacionales

De acuerdo con la empresa Oracle [8], las bases de datos relacionales son un medio de almacenamiento que permite la lectura y escritura de datos relacionados unos con otros. Se basan en un modelo relacional representado por tablas interconectadas por medio de un atributo único denominado clave.

Arquitectura por capas

Es un patrón de arquitectura basado en capas el cual alcanza independencia y separación entre las funcionalidades de cada una. En la Figura 1 se puede apreciar su diseño genérico. [9]

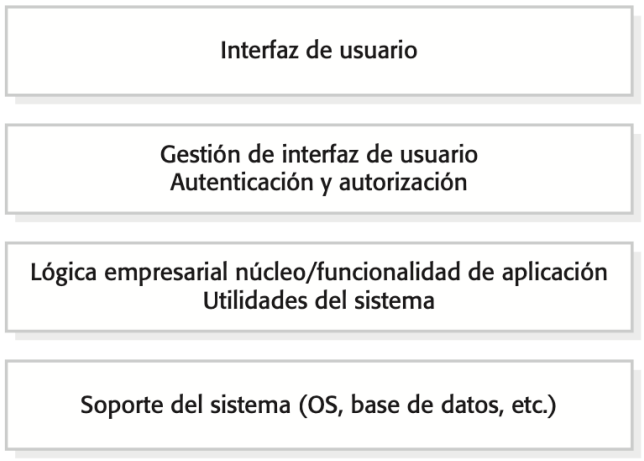


Figura 1 Arquitectura genérica por capas [9]

Las capas se apoyan en los servicios que le ofrece su capa inmediata inferior. Entre sus principales beneficios está el soporte para un desarrollo por incrementos, las capas son portátiles mientras las interfaces definidas inicialmente no sufran cambios.

En el cambio de interfaz entre capas solo se ve afectada las capas dependientes y a medida que los sistemas tienen dependencias en las capas inferiores se facilita la implementación de aplicaciones multiplataforma sobre la misma, es posible que capas que comprendan sistemas operativos o bases de datos tengan que ser reimplantadas para su facilidad, pero esto no debería tener mayor impacto en capas superiores. [9]

Herramientas de desarrollo

La herramienta de desarrollo y su descripción se encuentran en la Tabla 1.

Tabla 1 Herramientas utilizadas para el desarrollo del componente

Herramienta	**Descripción**
Visual Studio Code	Es un editor de código fuente de código abierto disponible para macOS, Windows y Linux. Mayormente usado para desarrollo de aplicaciones web y para casi cualquier lenguaje de programación. [10]
PGAdmin4	Es una plataforma de desarrollo y gestión de bases de datos PostgreSQL de código abierto, disponible para macOS, Windows, Unix y Linux. [11]
Notion	Es una herramienta útil para la gestión de proyectos, planificación y tomar notas. Permite el trabajo colaborativo. [12]
Gitlab	Esta herramienta de código abierto permite el control de versiones con Git, rastreo de problemas, revisión de código entre otras funcionalidades útiles para el desarrollo de software. [13]
Postman	Plataforma destinada para la construcción y uso de APIs. Permite el trabajo colaborativo. [14]

Tecnologías

Las tecnologías de desarrollo se describen en la Tabla 2.

Tabla 2 Tecnologías de desarrollo

Tecnología	**Descripción**
PostgreSQL	Es un gestor de bases de datos de tipo relacionales que soporta una gran cantidad de datos está totalmente orientado a objetos, es de código abierto, lo que lo

	hace totalmente gratuito, tiene una fuerte comunidad que se encarga de mantener y actualizar constantemente de acuerdo con los requerimientos tecnológicos emergentes, la sintaxis SQL que usa está basada en el estándar ISO/IEC SQL por lo que es posible la comunicación con otras bases de datos. Además, cumple con los principios ACID (Atomicidad, Consistencia, Asilamiento y Durabilidad). Dentro de sus componentes está la integración de triggers, herencia de tablas, uso de esquemas en una misma base y procedimientos almacenados optimizados. Adicionalmente se incorpora la creación de procedimientos almacenados en Python y PERL. [15]
Node JS	Node JS es un entorno de ejecución de JS, permitiéndonos ejecutar JS fuera del navegador e incorporarlo como un lenguaje normal como si se tratase de Java o C++, con ello podemos tener el poder JS para el diseño, implementación y testing para cualquier tipo de aplicación. Está diseñado para crear aplicaciones altamente escalables ya que facilita la gestión de múltiples conexiones debido a que maneja el concepto de asincronismo, el cual usa el evento loop con lo cual se puede procesar varias peticiones de forma asíncrona. [16]
Express	Express es un framework para Node JS que permite la creación de aplicaciones y de infraestructura orientadas a la web, se caracteriza por ser minimalista, permite

	hacer tanto aplicaciones web, servir páginas tanto estáticas como no estáticas ya sean en tecnologías nativas o con frameworks de frontend, adicionalmente permite la creación de APIs REST, para ello nos facilita todos los métodos HTTP que se tienen disponibles en el protocolo, así como la disposición de elementos comunes como los middlewares, permitiendo así la creación de endpoints de forma fácil y sencilla. Tiene un alto rendimiento debido a que está basado en las características principales de Node JS, por lo que esta tecnología solo extiende los métodos HTTP sin dejar de lado las bondades que ya trae Node JS. [17]
Bcrypt	Es una librería JS que permite el hash de contraseñas de forma rápida y sencilla, contiene una configuración que permite la incorporación de un salt, un método generador del hash y finalmente un método que permite comparar contraseñas. [18]

2 METODOLOGÍA

2.1 Análisis y planificación del proyecto

Previo a la realización de este proyecto de software se debe realizar un análisis previo considerando las ideas inicialmente expuestas, limitantes del proyecto, expectativas, alcance, tiempo, presupuesto y funcionalidades contempladas y se espera que el sistema pueda cumplir. Con estas generalidades se debe analizar y comparar para definir las bases que guían el desarrollo y cumplimiento del proyecto, estas son: escoger una metodología de desarrollo de software, una arquitectura que se ajuste a las necesidades, tecnologías a ser usadas, definir una infraestructura para la implantación del producto final y el proceso de implementación o codificación.

2.2 Metodología de desarrollo

El éxito en un proyecto de software depende en gran medida de la metodología escogida, por lo tanto, antes de seleccionar una metodología en concreto, lo primero que se realizó es escoger un grupo o enfoque de metodologías que se acoplen a las características del proyecto formulado y las necesidades de LAB-FIS, en específico existen dos enfoques, el tradicional y ágil.

Metodologías de desarrollo tradicionales

En cuanto a las metodologías de desarrollo tradicionales se caracterizan por ser metodologías clásicas que necesitan tener los requerimientos bien definidos y estables, es decir requerimientos que no van a cambiar con el tiempo, se debe tener una idea muy clara del producto final, en otras palabras, la lista precisa de necesidades que se requieren, no basta con tener ideas generales del producto. Se centran en el cumplimiento del plan del proyecto y la generación de una documentación exhaustiva, por lo general no admiten cambios durante el proceso de desarrollo, en general son lineales ya que de una forma u otra realizan las fases de requerimientos, diseño, implementación y pruebas.

Si bien existen variaciones como algunas basadas en prototipos o iteraciones, estas no dejan de ser lineales ya que cada iteración es un pequeño proyecto lineal que no admite cambios. En caso de realizar cambios estos resultan ser muy costosos, finalmente están centrados en procesos previamente definidos. Entre las metodologías de desarrollo que se encuentran en este grupo tenemos a cascada, RUP, espiral, Microsoft Solution Framework (MSF) y prototipos.

Metodologías de desarrollo ágiles

En cuenta a las metodologías de desarrollo ágiles se caracterizan por ser un enfoque basado en el agilismo implementado en el mundo del desarrollo de software, con esto en mente, se permite tener un inicio de la implementación más rápida, comunicación y retroalimentación activa de los interesados en el proyecto para poder hacer adaptaciones al producto, se pueden hacer cambios a lo largo del proyecto sin elevar demasiado el costo de desarrollo, se pueden retrasar algunas decisiones, planificación adaptativa a las circunstancias, aunque en un principio podría parecer un proceso informal y poco

controlado ya en la práctica las metodologías ágiles tienen todo bien definido y por su enfoque incremental se adaptan perfectamente a proyectos a gran escala, en los que es imposible saber con máximo detalle todos los requerimientos y necesidades a ser cubiertas, pero si se dispone de una idea general, en estos casos las metodologías ágiles resultan ser la solución perfecta, pues se adaptan al avance del proyecto.

Se considera que las interacciones con los involucrados y expresar sus necesidades es más importante que las herramientas y los procesos, lo importante es un producto de calidad antes que una exhaustiva documentación y una respuesta inmediata y bien recibida frente a los cambios. Entre las metodologías que se encuentran en este grupo tenemos a SCRUM, XP, Cristal y Kanban.

Selección de un enfoque para el proyecto

Para el desarrollo del CORE se ha realizado la siguiente comparación, en la que exponemos las principales características de este proyecto y se marca con una X la metodología que mejor se adapta, para al final escoger un enfoque ágil o tradicional, todo esto descrito en la Tabla 3.

Tabla 3 Comparación de metodologías [Autoría propia]

Característica	Metodologías de enfoque ágil	Metodologías de enfoque tradicional
Requerimientos no están bien definidos, si bien nos han expresado las intenciones que se tiene para construir un CORE que permita la gestión de los procesos que se llevan a cabo en LAB-FIS, no se tienen muy claro exactamente las operaciones que involucrará cada módulo.	X	
Documentación, en este apartado se nos ha especificado que no es necesario una documentación exhaustiva del desarrollo, fundamentalmente se necesitarían manuales del programador para poder hacer cambios de ser necesario, estos manuales no deben ser extremadamente detallados y adicionalmente se	X	

necesitaría una documentación de los servicios del CORE para que futuros trabajos se puedan adaptar fácilmente.		
Reuniones con los involucrados, se requieren reuniones periódicas para poder verificar y validar el avance que tiene el CORE, así como poder dar retroalimentación, correcciones, cambios y definir bien el alcance del siguiente módulo del proyecto.	X	
Planificación y cronograma, en este apartado se tiene claro que la planificación es muy rigurosa y poco flexible debido a que el producto final debe ser desarrollado e implantado en el tiempo que dura el semestre académico y no es posible extensiones de tiempo.		X

Como resultado de esta comparación podemos ver que el enfoque adecuado es un enfoque ágil.

Selección de una metodología ágil

Para el desarrollo del CORE se ha seleccionado directamente a SCRUM debido a que el equipo de desarrollo y los involucrados tienen amplia experiencia tanto en el ámbito académico como laboral, por lo tanto, se tiene conocimiento teórico y práctico que facilitan la implementación de esta metodología. Pero adicionalmente se ha realizado la siguiente lista de chequeo que permite validar si SCRUM se adapta al proyecto actual (Tabla 4).

Tabla 4 Lista de chequeo de metodología SCRUM [Autoría propia]

Característica	**Metodología SCRUM**
Enfoque ágil	Sí
Revisiones constantes y retroalimentación	Sí, mediante los Sprint Reviews
Conocimiento y experiencia de la metodología	Sí, por parte de todo el equipo y adicionalmente por parte de los involucrados.
Entrega continua de productos funcionales	Sí

En conclusión, la metodología escogida es SCRUM debido a su enfoque ágil, su completa adaptación de acuerdo con las características de este proyecto y la experiencia del equipo usando esta metodología.

2.3 Ingeniería de requerimientos

Entrevistas de usuario

En este proyecto la técnica seleccionada para poder obtener los requerimientos de un usuario es mediante entrevistas guiadas, para ello se tiene un pequeño guion de las preguntas que como desarrolladores y gestores del proyecto son útiles, sin embargo, siempre se concedió total libertad a los entrevistados para que puedan expresar ideas adicionales, inconvenientes, ejemplos o posibles soluciones, siguiendo este patrón se han desarrollado dos entrevistas de una hora y media cada una para poder obtener el producto backlog, posteriormente en cada iteración en el período del Sprint Planning se mantuvo una nueva reunión de una hora con la finalidad de poder entrar a mayor detalle en cada uno de los módulos en específico y poder definir con exactitud los campos y demás especificaciones necesarias.

User Story Mapping

User Story Mapping es una técnica de visualización y organización de historias de software del software a fin de que nos permitan un manejo eficiente y ágil de los requerimientos expresados. Consta de los siguientes componentes:

- Usuario: Es una pequeña historia del usuario final del sistema o de la parte del sistema, se incluye información acerca del mismo y sus tareas que realiza de forma cotidiana.
- Backbone: Sección del User Story Map constituido por objetivos y actividades
- Objetivos: Meta a cumplir que tiene el usuario y el sistema debe automatizar o ayudar
- Actividades: Unidades de trabajo que en conjunto ayudan a cumplir los objetivos propuestos.

Considerando esto y las necesidades expresadas en las reuniones y entrevistas con el personal de LAB-FIS se ha generado el siguiente User Story Map que puede ser observado en el Anexo I

Roles de SCRUM

De acuerdo con la metodología seleccionada se definieron los tres roles necesarios, se describen en la Tabla 5.

Tabla 5 Roles de SCRUM

Rol	**Responsable**
Scrum Máster	Anderson Cárdenas
Product Owner	Ing. Rodrigo Chancusig, Msc
Equipo de desarrollo	Javier Erazo y Anderson Cárdenas

2.4 Planificación del proyecto

Tabla 6 Planificación del proyecto

	Semana 1 y 2 Sprint 1	**Semana 3 y 4 Sprint 2**	**Semana 5 y 6 Sprint 3**	**Semana 7 y 8 Sprint 4**	**Semana 9 y 10 Sprint 5**
Inventario					
Préstamos					
Servicios					
Asistencias					
Autenticación y autorización					

Considerando las restricciones del proyecto inherentes a la planificación académica de un semestre, la priorización basada en los requerimientos y la metodología seleccionada se determinó que cada Sprint o iteración tenga una duración de dos semanas, respetando los principios de la metodología SCRUM, la planificación se muestra en la Tabla 6.

2.5 Análisis de arquitectura

A partir de los requerimientos detallados por el personal de LAB-FIS se procede a realizar el diseño arquitectónico, para lo cual se realiza una comparación de los patrones arquitectónicos que aplican en este caso. Como se detalla en la Tabla 7.

Tabla 7 Comparación de patrones arquitectónicos de acuerdo con [9]

Patrón Arquitectónico	Cuando usarlo	Ventajas	Desventajas
Arquitectura en Capas	En la construcción de software sobre sistemas existentes, cuando se cuentan con equipos de desarrollo dispersos con responsabilidades específicas y especialmente cuando se necesita seguridad en cada nivel.	Ayuda en la descomposición y mantenibilidad. Permite implementar diferentes funcionalidades redundantes. Soporta el desarrollo incremental.	El rendimiento puede resultar afectado cuando existe un gran crecimiento de capas. La descomposición limpia entre capas suele ser utópica.
Arquitectura de Repositorio	En sistemas que requieren manejar altos volúmenes de datos con un largo período de almacenamiento. En sistemas dirigidos por datos.	Se puede obtener independencia entre los componentes. Al tener los datos centralizados, se los puede gestionar consistentemente.	Afecta a la seguridad, debido a los datos centralizados se tiene un único punto de fallo. Puede existir ineficiencia al coordinar la comunicación entre todos los componentes.
Arquitectura Cliente Servidor	Cuando los datos se ingresan desde diferentes	Los diferentes servidores pueden	Los diferentes servicios con un único punto de

	puntos a un único medio de almacenamiento. Cuando se requiere independencia entre los servicios.	estar distribuidos por la red. Centraliza los servicios para que los clientes puedan acceder con facilidad.	fallo y son susceptibles a denegación de servicios o fallos del servidor.
Arquitectura de Tubería y Filtro	Cuando se realiza un procesamiento de datos en lotes o en transacciones por diferentes etapas relacionadas	Es comprensible y se adapta fácilmente a los procesos del negocio. Agregar nuevos componentes resulta sencillo debido a su secuencialidad.	Es necesario acordar transferencia de datos en cada comunicación. Los formatos de los datos pueden resultar en la imposibilidad de reutilizar componentes funcionales.

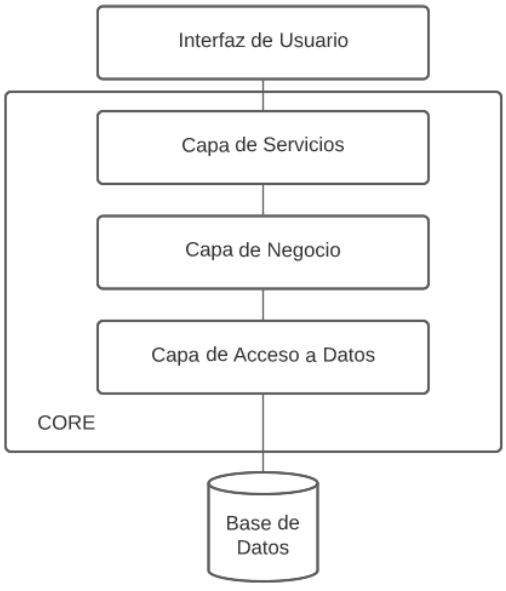


Figura 2 Arquitectura de alto nivel [Autoría propia]

Debido a que la prioridad en LAB-FIS es la mantenibilidad, la descomposición y la seguridad se escogió una arquitectura por capas. En la Figura 2 se describe la arquitectura de alto nivel.

Para el CORE se ha destinado tres capas principales:

La capa de servicios contiene todas las operaciones a las que puede acceder los distintos clientes y es el punto de entrada de estos, además, envía a la capa de negocio la información proporcionada por usuario.

La capa de negocio contiene la lógica de este, donde se realizan validaciones de los datos y especificación de sus modelos, implementaciones y validaciones de seguridad en cada una de las operaciones permitidas en la capa de servicios.

Finalmente, la capa de acceso a datos realiza el procesamiento dentro de las tablas de la base de datos y es capaz de retornar la información que ha sido solicitada por la capa de negocio.

Dado que se ha contemplado realizar el desarrollo para cuatro de los principales procesos de los laboratorios se diseñó una arquitectura a un nivel de abstracción superior, la cual se muestra en la Figura 3.

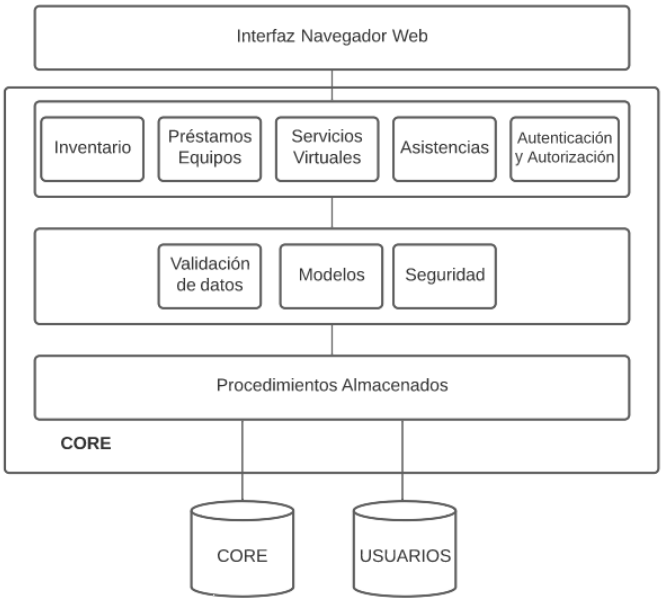


Figura 3 Segunda aproximación de arquitectura de alto nivel [Autoría propia]

Como se observa en la Figura 3 la capa de acceso a datos abarca el procesamiento de los datos mediante procedimientos almacenados. De acuerdo con [19] los procedimientos

almacenados proveen mayor seguridad, eficiencia y provee una ocultación de la información con el resto de la arquitectura, estas características son compatibles con los requerimientos no funcionales solicitados, por tanto, se ha decidido implementar como última capa de la arquitectura CORE, siendo esta capa la más importante.

2.6 Justificación de tecnologías

Lenguaje de Backend

Para seleccionar el lenguaje de programación de la API y la base de datos se tomó como criterio principal el no incurrir en software propietario, por lo tanto, se realizó una comparación entre los lenguajes de programación más populares y de código abierto de acuerdo con la empresa de software Jetbrains [20].

Para esto se han tomado en cuenta cinco aspectos principales, como se observa en la Tabla 8 (el mejor lenguaje se marca con una X).

Tabla 8 Comparación lenguajes de programación de backend basado en [21]

Lenguaje	**Rapidez de desarrollo**	**Eficiencia**	**Lean Backend**	**Entrada/Salida**	**Fácil aprendizaje**
JavaScript		X	X	X	
Python					X
Ruby	X				

Por lo tanto, el lenguaje de programación seleccionado es JavaScript, el cual con su entorno de ejecución Node JS serán usados para el desarrollo de la API.

Base de Datos

La base de datos que se tomó en cuenta para la comparación debe cumplir con las condiciones de ser de código abierto y relacional, para lo cual se tomaron tres de las más populares, como se observa en la Tabla 9 (se marca con X si cumple).

Tabla 9 Comparativas bases de datos de acuerdo con [22]

Base de Datos	**Alto Volumen de Datos**	**Concurrencia**	**Basada en la comunidad**	**Compatibilidad con NodeJS y arquitectura**	**Fácil aprender**
PostgreSQL	X	X	X	X	X
MariaDB	X		X	X	
MySQL	X				X

Luego de este análisis se determina que la mejor opción es PostgreSQL, principalmente por su rendimiento y gran cantidad de documentación disponible en internet, permite programar procedimientos almacenados y manejar concurrencia.

2.7 Preparación para desarrollo

Antes de empezar con el proceso de desarrollo se realizaron las siguientes actividades para aportar con la calidad del producto.

Estándar de desarrollo

Antes de comenzar con el desarrollo y para contribuir al atributo de calidad de la mantenibilidad, se realizó la definición del estándar de desarrollo que regirá en el actual desarrollo y para futuras implementaciones y modificaciones. El estándar a detalle se encuentra en el Anexo ll.

Entorno de desarrollo y pruebas

Para el entorno de desarrollo se utilizaron las siguientes herramientas de código abierto y de uso gratuito, se detallan en la Tabla 10.

Tabla 10 Herramientas utilizadas para el desarrollo

Herramienta	**Versión**	**Uso**
Visual Studio Code	1.70.1	Desarrollo de API

Node JS	17.3.1	Entorno de ejecución para la programación del REST Server
PostgreSQL	13.0	Motor de base de datos
Express	4.18.1	Framework para aplicaciones web backend
Gitlab	14.0	Gestión de control de versiones
Jest	28.1.3	Ejecución de pruebas unitarias
PGAdmin4	6.8	Administración y desarrollo de base de datos
Notion	2.0.24	Gestión de tareas
Supabase	Versión web	Almacenamiento de bases de datos para desarrollo
Postman	9.27.0	Gestor de peticiones para probar la API durante el desarrollo

2.8 Ejecución de los sprints y sus iteraciones

A continuación, se describe el proceso realizado en los cinco sprints planificados para el desarrollo.

Ejecución de Sprints para el módulo de inventario

En esta iteración se contemplaron las dos historias de usuario con mayor prioridad (Figura 5 y Figura 6)

Código	Título	Descripción	Prioridad	Estimación
HU01	Responsables de los equipos	*Como* usuario del sistema, quiero conocer quien(es) es(son) los responsables de cada uno de los equipos registrados en el inventario, para control y supervisión de los equipos	Alta	5

Figura 4 Historia de usuario Gitlab HU01

Código	Título	Descripción	Prioridad	Estimación
H02	Inventario de los equipos	*Como* usuario del sistema, quiero conocer toda la información de cada equipo así como el historial de sus encargados para control y supervisión.	Alta	5

Figura 5 Historia de usuario Gitlab HU02

Sprint Backlog

En base a las dos historias de usuario contempladas se obtuvieron las tareas detalladas en la Tabla 11.

Tabla 11 Tareas para el módulo de inventario

Código	Descripción	Responsable
TA01	Diseño e implementación base de datos equipos	Javier Erazo
TA02	Procedimiento almacenado que permita: • Registrar un nuevo equipo de tipo computador • Registrar un nuevo equipo genérico • Obtener todos los equipos activos • Obtener la información del equipo a partir de su identificador • Cambiar el estado de un equipo • Buscar equipos a partir de su ubicación • Cambiar la ubicación de un equipo • Obtener el historial de ubicaciones de un equipo	Javier Erazo
TA03	Implementación de servicios en API que permitan: • Registrar un nuevo equipo de tipo computador • Registrar un nuevo equipo genérico • Obtener todos los equipos activos • Obtener la información del equipo a partir de su identificador	Anderson Cárdenas

	• Cambiar el estado de un equipo • Buscar equipos a partir de su ubicación • Cambiar la ubicación de un equipo Obtener el historial de ubicaciones de un equipo	
TA04	Diseño e implementación de base de datos de custodios	Javier Erazo
TA05	Procedimiento almacenado que permita: • Registrar un custodio • Obtener los custodios activos • Obtener un custodio a partir de su identificador • Cambiar el estado de un custodio	Javier Erazo
TA06	Implementación de servicios en API que permitan: • Registrar un custodio • Obtener los custodios activos • Obtener un custodio a partir de su identificador • Cambiar el estado de un custodio	Anderson Cárdenas
TA07	Procedimiento almacenado que permita: • Obtener los equipos asociados al custodio • Obtener el custodio asociado a un equipo • Asignar un equipo a un custodio	Javier Erazo
TA08	Implementación de servicios en API que permitan: • Obtener los equipos asociados al custodio • Obtener el custodio asociado a un equipo • Asignar un equipo a un custodio	Anderson Cárdenas

Ejecución del sprint

Luego de haber completado todas las tareas se obtuvo los entregables que se detallan en la Tabla 12.

Tabla 12 Resumen entregables ejecución del sprint 1

Entregable	**Número de elementos**
Base de datos	15 tablas
Procedimientos Almacenados	15 procedimientos
Servicios API	15 servicios

Los detalles de los entregables conseguidos en esta iteración se encuentran en el Anexo III y VIII.

Sprint Review

Para realizar el sprint review del módulo se convocó a dos representantes de LAB-FIS y al product owner. Se realizó la respectiva validación con los criterios de aceptación que se muestran en la Tabla 13, con un enfoque de cumplimiento o no cumplimiento.

Tabla 13 Resultados de sprint review en el módulo de inventario

Criterio de aceptación	**Responsables**	**Cumple / Parcialmente / No cumple**
¿Existe la posibilidad de gestionar los datos relacionados a los bienes de forma clara y concisa?	Ing. Rodrigo Chancusig, Msc, Ing. Jorge Miño, Ing. Milena Nazamues	Cumple
¿Existe la posibilidad de gestionar los datos relacionados a los custodios de los bienes de forma clara y concisa?	Ing. Rodrigo Chancusig, Msc, Ing. Jorge Miño, Ing. Milena Nazamues	Cumple
¿Las entidades cumplen con los datos necesarios para acoplarse a los procesos?	Ing. Rodrigo Chancusig, Msc, Ing. Jorge Miño, Ing. Milena Nazamues	Cumple
¿Los entregables están completos y son útiles para los laboratorios?	Ing. Rodrigo Chancusig, Msc, Ing. Jorge Miño, Ing. Milena Nazamues	Parcialmente

Cómo se observa en la Tabla 13 hubo un cumplimiento parcial en el último criterio, esto debido a que existieron dos cambios que se detallan en la Tabla 14.

Tabla 14 Cambios del sprint review en el módulo inventario

Responsable	**Descripción del cambio**
Ing. Jorge Miño	La gestión de los equipos se las debe hacer con el código del bien actual y el código de bien anterior

Ing. Jorge Miño	En el historial de ubicaciones del equipo se debe dar un detalle completo de ubicación y fecha

Ejecución de Sprints para el módulo de préstamos de equipos

En esta iteración se contempló una historia de usuario con mayor prioridad (Figura 7)

Código	Título	Descripción	Prioridad	Estimación
HU03	Préstamo de equipos	*Como* usuario del sistema, quiero llevar un registro de préstamo y devolución de los equipos tanto para docentes o estudiantes, para poder consolidar una lista negra de usuarios	Alta	3

Figura 6 Historia de usuario Gitlab HU03

En base a la historia de usuario HU03 y los cambios existentes en el anterior sprint se obtuvieron las tareas detalladas en la Tabla 15.

Tabla 15 Tareas para el módulo de préstamo de equipos

Código	Descripción	Responsable
TA09	Modificación de procedimientos almacenados para identificarlo por código anterior y el código actual del bien	Javier Erazo
TA10	Modificación de procedimientos almacenados en la búsqueda del historial del equipo	Anderson Cárdenas
TA11	Diseño e implementación base de datos de usuarios de préstamos	Javier Erazo
TA02	Procedimiento almacenado que permita: • Registrar un nuevo usuario de préstamos • Actualizar los datos de un usuario de préstamos • Obtener todos los usuarios de préstamos activos • Obtener la información del equipo a partir de su identificador • Cambiar el estado de un usuario de prestamos • Buscar un usuario de préstamos a partir de su cédula	Javier Erazo
TA03	Implementación de servicios en API que permitan: • Registrar un nuevo usuario de préstamos • Actualizar los datos de un usuario de préstamos	Anderson Cárdenas

	• Obtener todos los usuarios de préstamos activos • Obtener la información del equipo a partir de su identificador • Cambiar el estado de un usuario de prestamos • Buscar un usuario de préstamos a partir de su cédula	
TA04	Diseño e implementación de base de datos del personal de LAB-FIS	Javier Erazo
TA05	Procedimiento almacenado que permita: • Registrar el personal • Obtener todo el personal activo • Obtener el personal a partir de su identificador • Cambiar el estado del personal	Javier Erazo
TA06	Implementación de servicios en API que permitan: • Registrar el personal • Obtener todo el personal activo • Obtener el personal a partir de su identificador • Cambiar el estado del personal	Anderson Cárdenas
TA07	Procedimiento almacenado que permita: • Registrar la entrega de un equipo a un usuario • Registrar la recepción de un equipo al personal • Obtener los atrasos y préstamos no devueltos de un usuario • Justificar los atrasos y no entregas de un usuario • Obtener los préstamos realizados en un rango de fecha • Obtener el estado de un usuario de préstamos	Javier Erazo
TA08	Implementación de servicios en API que permitan: • Registrar la entrega de un equipo a un usuario • Registrar la recepción de un equipo al personal • Obtener los atrasos y préstamos no devueltos de un usuario • Justificar los atrasos y no entregas de un usuario • Obtener los préstamos realizados en un rango de fecha • Obtener el estado de un usuario de préstamos	Anderson Cárdenas

Ejecución del sprint

Luego de haber completado todas las tareas se obtuvo los entregables que se detallan en la Tabla 16.

Tabla 16 Resumen entregables ejecución del sprint 2

Entregable	**Número de elementos**
Base de datos	8 tablas
Procedimientos Almacenados	15 procedimientos
Servicios API	15 servicios
Servicios módulo inventario	2 servicios

Los detalles de los entregables conseguidos en esta iteración se encuentran en el Anexo IV y VIII.

Sprint Review

Para realizar el sprint review del módulo se convocó a dos representantes de LAB-FIS y al product owner. Se realizó la respectiva validación con los criterios de aceptación que se muestran en la Tabla 17.

Tabla 17 Resultados de sprint review en el módulo de préstamo de equipos

Criterio de aceptación	**Responsables**	**Cumple / Parcialmente / No cumple**
¿Existe la posibilidad de gestionar los datos relacionados a los usuarios de préstamos de forma clara y concisa?	Ing. Rodrigo Chancusig, Msc, Ing. Jorge Miño, Ing. Milena Nazamues	Cumple
¿Existe la posibilidad de gestionar los datos relacionados a los retrasos y no entregas de forma clara y concisa?	Ing. Rodrigo Chancusig, Msc, Ing. Jorge Miño, Ing. Milena Nazamues	Cumple

¿Las entidades cumplen con los datos necesarios para acoplarse a los procesos?	Ing. Rodrigo Chancusig, Msc, Ing. Jorge Miño, Ing. Milena Nazamues	Cumple
¿Los entregables están completos y son útiles para los laboratorios?	Ing. Rodrigo Chancusig, Msc, Ing. Jorge Miño, Ing. Milena Nazamues	Cumple

Una vez validado los entregables no existieron cambios para el módulo de préstamos de equipos al igual que las correcciones del módulo de inventario.

Ejecución de Sprints para el módulo de solicitudes de servicios

En esta iteración se contempló una historia de usuario con mayor prioridad (Figura 8)

Código	Título	Descripción	Prioridad	Estimación
HU04	Solicitudes de Servicios Virtuales	*Como* usuario del sistema, quiero gestionar las solicitudes de servicios virtuales de estudiantes y docentes, para poder tomar acción lo más pronto posible y brindar un servicio eficiente a los usuarios del laboratorio	Media	3

Figura 7 Historia de usuario Gitlab HU04

Sprint Backlog

En base a la historia de usuario HU04 se obtuvieron las tareas detalladas en la Tabla 18.

Tabla 18 Tareas para el módulo de préstamo de servicios

Código	Descripción	Responsable
TA11	Diseño e implementación base de datos de solicitudes de servicios	Javier Erazo
TA02	Procedimiento almacenado que permita: • Registrar un nuevo estudiante al que se le pueda asignar un servicio a partir de la solicitud de un docente • Registrar un nuevo docente que pueda solicitar un servicio	Javier Erazo
TA03	Implementación de servicios en API que permitan:	Anderson Cárdenas

	• Registrar un nuevo estudiante al que se le pueda asignar un servicio a partir de la solicitud de un docente • Registrar un nuevo docente que pueda solicitar un servicio	
	Procedimiento almacenado que permita: • Solicitar un servicio de tipo aplicativo por parte de un docente para varios estudiantes • Solicitar equipos remotos para los estudiantes de la materia que dicta • Solicitar un VDI por parte de un docente • Solicitar un servidor por parte de un docente	
	Implementación de servicios en API que permitan: • Solicitar un servicio de tipo aplicativo por parte de un docente para varios estudiantes • Solicitar equipos remotos para los estudiantes de la materia que dicta • Solicitar un VDI por parte de un docente • Solicitar un servidor por parte de un docente	
TA05	Procedimiento almacenado que permita: • Registrar las credenciales de acceso remoto relacionados a un estudiante y cambie el estado de este • Registrar el cambio de estado de una solicitud realizada por un docente • Obtener todos los servicios activos • Registrar la baja de una solicitud asociada a un servicio • Registrar el alta de un servicio asociado a una solicitud	Javier Erazo
TA06	Implementación de servicios en API que permitan: • Registrar las credenciales de acceso remoto relacionados a un estudiante y cambie el estado de este • Registrar el cambio de estado de una solicitud realizada por un docente	Anderson Cárdenas

	• Obtener todos los servicios activos • Registrar la baja de una solicitud asociada a un servicio • Registrar el alta de un servicio asociado a una solicitud	

Ejecución del sprint

Luego de haber completado todas las tareas se obtuvo los entregables que se detallan en la Tabla 19.

Tabla 19 Resumen entregables ejecución del sprint 3

Entregable	**Número de elementos**
Base de datos	12 tablas
Procedimientos Almacenados	12 procedimientos
Servicios API	12 servicios

Los detalles de los entregables conseguidos en esta iteración se encuentran en el Anexo V y el Anexo VIII.

Sprint Review

Para realizar el sprint review del módulo se convocó a dos representantes de LAB-FIS y al product owner. Se realizó la respectiva validación con los criterios de aceptación que se muestran en la Tabla 20.

Tabla 20 Resultados de sprint review en el módulo de préstamo de servicios

Criterio de aceptación	**Responsables**	**Cumple / Parcialmente / No cumple**
¿Existe la posibilidad de gestionar los datos relacionados a los docentes y estudiantes implicados en el préstamo de servicios de forma clara y concisa?	Ing. Rodrigo Chancusig, Msc, Ing. Jorge Miño, Ing. Milena Nazamues	Cumple
¿Existe la posibilidad de gestionar los datos y el estado relacionados a las	Ing. Rodrigo Chancusig, Msc, Ing. Jorge Miño, Ing. Milena Nazamues	Cumple

solicitudes de servicios de forma clara y concisa?		
¿Las entidades cumplen con los datos necesarios para acoplarse a los procesos?	Ing. Rodrigo Chancusig, Msc, Ing. Jorge Miño, Ing. Milena Nazamues	Cumple
¿Los entregables están completos y son útiles para los laboratorios?	Ing. Rodrigo Chancusig, Msc, Ing. Jorge Miño, Ing. Milena Nazamues	Cumple

Una vez validado los entregables no existieron cambios para el módulo de préstamos de servicios.

Ejecución de Sprints para el módulo de asistencias técnicas

En esta iteración se contempló la historia de usuario HU05 que se puede observar en la Figura 9

Código	Título	Descripción	Prioridad	Estimación
HU05	Registros de Asistencias Técnicas	*Como* usuario del sistema, quiero llevar un registro de las asistencias técnicas solicitadas al personal del laboratorio, para poder identificar las más frecuentes y tomar acción sobre ellas	Media	3

Figura 9 Historia de usuario Gitlab HU05

Sprint Backlog

En base a la historia de usuario HU05 se obtuvieron las tareas detalladas en la Tabla 21

Tabla 21 Tareas para el módulo de asistencias técnicas

Código	Descripción	Responsable
TA01	Diseño e implementación base de datos de asistencias técnicas	Javier Erazo
TA02	Procedimiento almacenado que permita: • Registrar una asistencia especial • Registrar una asistencia técnica	Javier Erazo

TA03	Implementación de servicios en API que permitan: • Registrar asistencias técnicas y asistencias especiales considerando sus datos requeridos.	Anderson Cárdenas
TA04	Procedimiento almacenado que permita: • Cambiar el estado de una asistencia como “en proceso” • Cambiar el estado de una asistencia como “terminada”	Javier Erazo
TA05	Implementación de servicios en API que permitan: • Cambiar el estado de una asistencia entre “en progreso” y “terminado”	Anderson Cárdenas
TA06	Procedimiento almacenado que permita: • Obtener los registros de asistencias especiales entre dos fechas, estado de la asistencia y considerando un laboratorio en específico. • Obtener los registros de asistencias técnicas entre dos fechas especiales y considerando el estado de la asistencia.	Javier Erazo
TA07	Implementación de servicios en API que permitan: • Consultar las asistencias especiales basándonos en una fecha de inicio y en una fecha de fin, adicionalmente y de forma opcional se podría especificar el laboratorio en el que se han realizado estas asistencias.	Anderson Cárdenas

Ejecución del sprint

Luego de haber completado todas las tareas se obtuvo los entregables que se detallan en la Tabla 22.

Tabla 22 Resumen entregables ejecución del sprint 4

Entregable	**Número de elementos**
Base de datos	5 tablas
Procedimientos Almacenados	6 procedimientos
Servicios API	6 servicios

Los detalles de los entregables conseguidos en esta iteración se encuentran en el Anexo VI y el Anexo VIII

Sprint Review

Para realizar el sprint review del módulo se convocó a dos representantes de LAB-FIS y al product owner. Se realizó la respectiva validación con los criterios de aceptación que se muestran en la Tabla 23.

Tabla 23 Resultados de sprint review en el módulo de asistencias técnicas

Criterio de aceptación	**Responsables**	**Cumple / Parcialmente / No cumple**
¿Existe la posibilidad de que las asistencias técnicas sean generales y permitan registrar diferentes situaciones?	Ing. Rodrigo Chancusig, Msc, Ing. Jorge Miño, Ing. Milena Nazamues	Cumple
¿Las asistencias especiales están orientadas a registrar los requerimientos especiales en los laboratorios de Lab-FIS?	Ing. Rodrigo Chancusig, Msc, Ing. Jorge Miño, Ing. Milena Nazamues	Cumple
¿Las entidades cumplen con los datos necesarios para acoplarse a los procesos?	Ing. Rodrigo Chancusig, Msc, Ing. Jorge Miño, Ing. Milena Nazamues	Cumple
¿Los entregables están completos y son útiles para los laboratorios?	Ing. Rodrigo Chancusig, Msc, Ing. Jorge Miño, Ing. Milena Nazamues	Cumple

Una vez validado los entregables no existieron cambios para el módulo de préstamos de servicios.

Ejecución de Sprints para el módulo de autenticación y autorización

En esta iteración se contempló la historia de usuario HU06 que se puede observar en la Figura 10

Código	Título	Descripción	Prioridad	Estimación
HU06	Autenticación y autorización de usuarios	*Como* administrador del sistema, quiero que mis usuarios tengan roles que permitan habilitar y restringir el acceso a cada módulo, para poder tener un control de autenticación y autorización.	Media	5

Figura 10 Historia de usuario Gitlab HU06

Sprint Backlog

En base a la historia de usuario HU06 se obtuvieron las tareas detalladas en la Tabla 24

Tabla 24 Tareas para el módulo de autenticación y autorización

Código	Descripción	Responsable
TA01	Diseño e implementación base de datos de usuarios y sus roles	Javier Erazo
TA02	Procedimiento almacenado que permita: • Obtener las credenciales (usuario y contraseña “hash”) dado el nombre de usuario. • Obtener los roles de usuario dado el nombre de usuario.	Javier Erazo
TA03	Implementación de servicios en API que permitan: • Hacer login de los usuarios registrados validando su contraseña mediante una comparación criptográfica del hash de la contraseña almacenada y contraseña ingresada, para que el usuario obtenga un JWT mismo que tiene su nombre de usuario.	Anderson Cárdenas
TA04	Implementación de middlewares en API que permitan: • Que cada petición deba llevar el JWT entregado en el login del usuario, el middleware se encarga de validar el JWT y si la firma es válida para el servidor web y adicionalmente validando los roles relacionados a las rutas de cada módulo y sus permisos de acceso	Anderson Cárdenas

Ejecución del sprint

Luego de haber completado todas las tareas se obtuvo los entregables que se detallan en la Tabla 25.

Tabla 25 Resumen entregables ejecución del sprint 5

Entregable	**Número de elementos**
Base de datos	3 tablas
Procedimientos Almacenados	2 procedimientos
Servicios API	1 servicios
Middlewares	5 middlewares

Los detalles de los entregables conseguidos en esta iteración se encuentran en el Anexo VII y el Anexo VIII

Sprint Review

Para realizar el sprint review del módulo se convocó a dos representantes de LAB-FIS y al product owner. Se realizó la respectiva validación con los criterios de aceptación que se muestran en la Tabla 26.

Tabla 26 Resultados de sprint review en el módulo de autenticación y autorización

Criterio de aceptación	**Responsables**	**Cumple / Parcialmente / No cumple**
¿Existe una forma de hacer login para cada usuario?	Ing. Rodrigo Chancusig, Msc, Ing. Jorge Miño, Ing. Milena Nazamues	Cumple
¿La comparación de las contraseñas se da mediante comparación de hash?	Ing. Rodrigo Chancusig, Msc, Ing. Jorge Miño, Ing. Milena Nazamues	Cumple
¿Se han agregado los middlewares necesarios para proteger cada módulo?	Ing. Rodrigo Chancusig, Msc, Ing. Jorge Miño, Ing. Milena Nazamues	Cumple

¿Se tiene un usuario admin que pueda acceder a todos los servicios de la API?	Ing. Rodrigo Chancusig, Msc, Ing. Jorge Miño, Ing. Milena Nazamues	Cumple
¿Las entidades cumplen con los datos necesarios para acoplarse a los procesos?	Ing. Rodrigo Chancusig, Msc, Ing. Jorge Miño, Ing. Milena Nazamues	Cumple
¿Los entregables están completos y son útiles para los laboratorios?	Ing. Rodrigo Chancusig, Msc, Ing. Jorge Miño, Ing. Milena Nazamues	Cumple

Una vez validado los entregables no existieron cambios para el módulo de autorización y autenticación.

2.9 Documentación

Al terminar con la ejecución de los sprints, el personal de los laboratorios requirió una documentación apropiada que ayude en las futuras implementaciones sobre la arquitectura CORE desarrollada en el presente trabajo, para satisfacer este requerimiento se desarrolló la documentación del API REST con un documento de Swagger (Anexo IX) y el manual del programador (Anexo X)

2.10 Pruebas unitarias

Con la ayuda de la librería Jest de JavaScript se realizaron pruebas unitarias de acuerdo con las operaciones principales de cada uno de los módulos. Las operaciones que se tomaron en cuenta se encuentran en la Tabla 27.

Tabla 27 Operaciones a ser probadas

Operación	**Módulo**
Cambiar estado del custodio	Inventario
Asignar un equipo a un custodio	Inventario
Buscar equipos por ubicación	Inventario
Cambiar estado de usuario de préstamos	Préstamos de equipos

Registrar entrega equipo	Préstamos de equipos
Registrar recepción equipo	Préstamos de equipos
Obtener problemas de préstamos	Préstamos de equipos
Cambiar a lista gris al usuario	Préstamos de equipos
Obtener la lista del usuario	Préstamos de equipos
Solicitar servicio aplicativo	Préstamo de servicios
Solicitar servicio de equipos remotos	Préstamo de servicios
Solicitar servicio de VDI	Préstamo de servicios
Solicitar servicio servidor	Préstamo de servicios
Obtener asistencias técnicas especiales	Asistencias técnicas
Obtener asistencias técnicas	Asistencias técnicas

El nivel de cobertura y lo obtenido en este apartado se muestra en la sección de resultados.

2.11 Despliegue

Escoger una infraestructura de despliegue para la implantación del producto final es importante para poder conocer las herramientas que utilizaremos y las características mínimas que se requieren para el correcto funcionamiento del CORE. Actualmente se tiene dos alternativas que son los servicios en la nube (también conocidos como servicios autogestionados) y por otro lado está un despliegue usando servidores propios. En cuanto a la primera alternativa, necesariamente se requiere de un presupuesto y eventualmente invertir un poco más si el número de usuarios crece y se necesita un escalamiento horizontal o vertical lo que implicaría un mayor gasto. LAB-FIS no dispone de este presupuesto por lo que la alternativa escogida es usar nuestros propios servidores. Considerando lo expuesto se tiene y las tecnologías escogidas se tiene la siguiente infraestructura mostrada en la Figura 11.

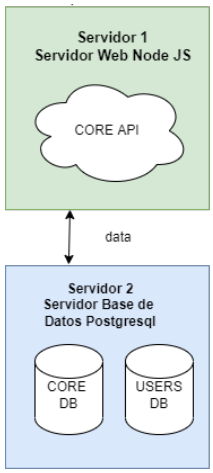


Figura 11 Infraestructura de dos servidores para el CORE [Autoría propia]

Como se puede apreciar se utilizarán dos servidores, uno dedicado a la recepción y respuesta de peticiones HTTP y el otro dedicado a las dos bases de datos utilizadas en el CORE. Las características mínimas requeridas para los servidores se observan en la Tabla 28.

Tabla 28 Requerimientos para el correcto funcionamiento del CORE [Autoría Propia]

Servidor	Función	Sistema Operativo	Memoria	CPU	Almacenamiento
Servidor 1	Web	Ubuntu Server 22.04	2 GB	1 CPU	20 GB
Servidor 2	Base de datos	Ubuntu Server 22.04	4 GB	1 CPU	40 GB

En caso de que en los próximos años se incremente de manera considerable la cantidad de usuarios se tendría que escalar ya sea de forma horizontal (agregando servidores con redundancia) o de forma vertical (mejorando las características del servidor). Adicionalmente al no ser servidores autogestionados eventualmente se requerirán optimizaciones, configuraciones y actualizaciones a nivel de sistema operativo.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 Resultados

Resultados de pruebas unitarias

Luego de haber terminado con la implementación de los cinco módulos planificados, a pesar de que todas las rutas han sido probadas y evaluadas tanto por parte del equipo de desarrollo como por los usuarios finales, se procedió a realizar tests unitarios más

detallados para las peticiones críticas, considerando que el 100% de cobertura en testing es imposible recurrimos a una priorización de las rutas más críticas para el correcto funcionamiento de los procesos de LAB-FIS. Estas han sido probadas usando la librería jest y la conexión a la base de datos, a continuación, se detalla las rutas probadas respecto al total del número de rutas existentes en cada uno de los módulos.

Módulo Inventario

Rutas probadas: 4 / 15 (Tabla 29)

Tabla 29 Resultados rutas críticas del módulo inventario

Petición	**Ruta**	**Consideraciones Adicionales**	**Resultado**
PATCH	/custodios/cambiar_estado_custodio	Se verifica el cambio de custodio en la base de datos	Correcto
PATCH	/equipos/asignar_equipo_custodio	Se verifica la asignación del custodio en la base de datos	Correcto
GET	/equipos/buscar_equipos_ubicacion	Se verifica que se obtenga el edificio, piso y aula en la que se encuentra el equipo considerando el formato actual de la EPN	Correcto
PATCH	/equipos/cambiar_ubicacion_equipo	Se verifica que se realice el cambio, pero adicionalmente que se registre el cambio en el historial de movimientos.	Correcto

Módulo Préstamos

Rutas probadas: 6 / 15 (Tabla 30)

Tabla 30 Resultados rutas críticas del módulo préstamo de equipos

Petición	**Ruta**	**Consideraciones Adicionales**	**Resultado**
PATCH	/usuarios_prestamos/cambiar_estado _usuario_prestamos	Se verifica que el usuario no tenga retrasos en la entrega de sus equipos	Correcto
PATCH	/prestamos/registrar_entrega_equipo	Ninguna, se realizan varios casos	Correcto
PATCH	/prestamos/registrar_recepcion_equipo	Ninguna, se realizan varios casos	Correcto
GET	/prestamos/obtener_problemas_prestamos _cedula	Ninguna, se realizan varios casos	Correcto
PATCH	/prestamos/cambiar_lista_gris_ usuario_prestamo	Se verifica que se registre en la tabla de incidentes y que se justifiquen los retrasos	Correcto
GET	/prestamos/obtener_lista_usuario	Ninguna, se realizan varios casos	Correcto

Módulo Servicios

Rutas probadas: 6 / 14 (Tabla 31)

Tabla 31 Resultados rutas críticas del módulo préstamo de servicios

Petición	**Ruta**	**Consideraciones Adicionales**	**Resultado**
POST	/solicitar_servicio_aplicativo	Ninguna, se realizan varios casos	Correcto

POST	/solicitar_servicio_equipo_remoto	Ninguna, se realizan varios casos	Correcto
POST	/solicitar_servicio_vdi	Ninguna, se realizan varios casos	Correcto
POST	/solicitar_servicio_servidor	Ninguna, se realizan varios casos	Correcto
PATCH	/bajar_servicio_activo	Ninguna, se realizan varios casos	Correcto
PATCH	/activar_solicitud_servicio	Ninguna, se realizan varios casos	Correcto

Módulo Asistencias

Rutas probadas: 2 / 6 (Tabla 32)

Tabla 32 Resultados rutas críticas del módulo asistencias

Petición	**Ruta**	**Consideraciones Adicionales**	**Resultado**
GET	/obtener_asistencias_especiales	Verificar varios casos incluyendo estado, fecha de inicio, fecha de fin y laboratorio	Correcto
GET	/obtener_asistencias_tecnicas	Se verifica usando varios filtros de fechas	Correcto

Como se puede observar en esta sección se ha realizado un testing más exhaustivo en aquellas operaciones que se consideran críticas, con esto se tiene un total de 18 rutas probadas con mayor detalle del total de la API que son un total de 50 rutas, es decir tenemos una cobertura de 36% para la API, se han definido 18 operaciones críticas para el negocio y de estas se han probado las 18, teniendo una cobertura del 100% en rutas críticas.

Resultados de pruebas del ciclo del negocio

Luego de haber implementado los módulos solicitados y considerando los requerimientos iniciales tenemos la siguiente comparación realizada de manera conjunta con el equipo de desarrollo y todos los involucrados, los resultados se aprecian en la Tabla 33.

Tabla 33 Resultados pruebas del ciclo del negocio

Requerimiento inicial	Cumplimiento
¿Es posible la gestión de los equipos del inventario?	Sí cumple
¿Es posible la gestión de los custodios	Sí cumple
¿Es posible conocer la ubicación detallada de un equipo?	Sí cumple
¿Para la búsqueda de equipos se tiene en cuenta tanto el código de bien anterior como el código de bien actual?	Sí cumple
¿Es posible la gestión de préstamos considerando las fechas de entrega y recepción?	Sí cumple
¿Se considera la inclusión de lista blanca, lista gris y lista negra de los usuarios considerando el historial de sus préstamos?	Sí cumple
¿Es posible que se soliciten todos los tipos de servicios que se manejan en el laboratorio?	Sí cumple
¿Es posible el registro de asistencias especiales y asistencias técnicas?	Sí cumple

3.2 Conclusiones

Presenta lo novedoso del trabajo de integración curricular, así como evaluación del cumplimiento o no de lo propuesto en los objetivos. En el caso en que no se cumpla uno o varios objetivos, y no se logren los resultados esperados, se propone una posible respuesta que explique por qué sucedió esto o las falencias de la planteado.

El uso de una arquitectura por capas es el más apropiado después de haber realizado el respectivo análisis y comparación, satisface las necesidades de seguridad y escalabilidad en LAB-FIS. También permite una comprensión clara de la relación entre cada uno de los componentes y la comunicación que mantienen entre ellos.

El uso de entrevistas y user story mapping permitieron capturar las necesidades reales del personal de los laboratorios, esto permitió captar el problema que se buscaba resolver en cada uno de los procesos y ofrecer un acceso apropiado y oportuno a los datos que conlleva cada uno de estos.

Postgres fue el motor de base de datos seleccionado para este proyecto, dado que cumple con la seguridad requerida para el mismo, sobre todo beneficia a LAB-FIS por ser de código abierto. Por otro lado, permite la comunicación apropiada con la API y la programación procedimental requerida por la arquitectura previamente definida.

La API desarrollada en NodeJS permite el acceso y modificación de los datos acorde a la arquitectura por capas y el manejo apropiado de las estructuras de datos necesarias para dar paso al flujo de la información en cada tarea de los procesos de inventario, préstamo de equipos, préstamo de servicios, asistencias técnicas y sobre todo en la autenticación y a autorización basada en roles que es requerida por el personal de LAB-FIS.

Se utilizó la infraestructura de LAB-FIS para el despliegue del producto final desarrollado en este proyecto, permitió las configuraciones correspondientes dado que se instalaron dos servidores independientes. En conjunto con la configuración de acceso del firewall y habilitación de puertos se consiguió establecer el punto de entrada a la API.

Las pruebas unitarias permitieron realizar la verificación de las operaciones principales dentro de cada uno de los módulos. El uso de la librería Jest permite realizar pruebas automatizadas y pruebas de regresión en el caso de requerir cambios posteriores tanto en la API como en la base de datos.

3.3 Recomendaciones

A lo largo del proyecto se ha priorizado el uso de herramientas de código abierto y que ofrezcan características acordes al proyecto, por lo cual, para futuras implementaciones se recomienda siempre priorizar las herramientas que cumplan con esto.

El entendimiento de la arquitectura y la comunicación entre cada una de las capas es requerido antes de comenzar con el desarrollo de cualquier componente, ya que sin esto se pueden insertar errores en la estructura de los datos que se transfieren de la base de datos a la API.

Se recomienda realizar pruebas de rendimiento y seguridad sobre la infraestructura de LAB-FIS en la que se hizo el despliegue del producto funcional, con el propósito de conseguir una disponibilidad acorde a la carga de uso de los futuros sistemas.

Utilizar la capa gratuita de la plataforma en la nube Supabase para futuras modificaciones y desarrollos en la base de datos, ya que es compatible con Postgres y facilita el trabajo colaborativo.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] Víctor Mejía Henao, “La informática y su contribución a la automatización de procesos,” Aug. 06, 2018. https://revistas.ceipa.edu.co/index.php/lupa/article/download/385/431?inline=1 (accessed Aug. 28, 2022).

[2] M. I. Romero *et al.*, “INTRODUCCIÓN A LA SEGURIDAD INFORMÁTICA Y EL ANÁLISIS DE VULNERABILIDADES.”

[3] I. A. Dolores, R. Peña, I. Luis, G. Silva, and R. Ii, “Arquitectura de software para el sistema de visualización médica Vismedic,” *Rev. Cuba. Informática Médica*, vol. 8, no. 1, pp. 75–86, 2016, Accessed: Aug. 28, 2022. [Online]. Available: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1684-18592016000100006&lng=es&nrm=iso&tlng=es.

[4] B. Meyer, “Agile principles,” *Agile!*, pp. 49–78, 2014, doi: 10.1007/978-3-319-05155-0_4.

[5] G. Bartolomeo and T. Kovacikova, “Hypertext Transfer Protocol,” *Identif. Manag. Distrib. Data*, pp. 31–48, 2013, doi: 10.1201/b14966-5.

[6] World Wide Web Consortium, “Web of Services - W3C,” 2015. https://www.w3.org/standards/webofservices/ (accessed Aug. 14, 2022).

[7] S. Edition, *REST API Development with Node.js - Google Books*. .

[8] Oracle, “What Is a Relational Database | Oracle.” https://www.oracle.com/database/what-is-a-relational-database/ (accessed Aug. 14, 2022).

[9] Sommerville, *Ingeniería de software 9*. .

[10] Microsoft, “Visual Studio: IDE y Editor de código para desarrolladores de software y Teams,” 2022. https://visualstudio.microsoft.com/es/ (accessed Aug. 14, 2022).

[11] pgAdmin, “pgAdmin - PostgreSQL Tools.” https://www.pgadmin.org/ (accessed Aug. 14, 2022).

[12] Notion, “Notion for projects & tasks,” 2022. https://www.notion.so/projects (accessed Aug. 14, 2022).

[13] Gitlab, “GitLab.org / GitLab · GitLab,” 2022. https://gitlab.com/gitlab-org/gitlab

(accessed Aug. 14, 2022).

[14] Postman, “Postman API Platform | Sign Up for Free,” 2022. https://www.postman.com/ (accessed Aug. 14, 2022).

[15] AyudaLeyData, “Ayuda Ley Protección Datos,” May 06, 2020. https://ayudaleyprotecciondatos.es/bases-de-datos/que-es-postgresql-ventajas/ (accessed Aug. 28, 2022).

[16] NodeJS org, “Acerca | Node.js,” Apr. 03, 2017. https://nodejs.org/es/about/ (accessed Aug. 28, 2022).

[17] Express org, “Documentación Express,” May 06, 2018. https://expressjs.com/es/ (accessed Aug. 28, 2022).

[18] Bcrypt doc, “NPMJS Bcrypt, documentation and examples for JS hashing,” Feb. 06, 2021. https://www.npmjs.com/package/bcrypt (accessed Aug. 28, 2022).

[19] R. Dewson, *for $ Evelopers*. .

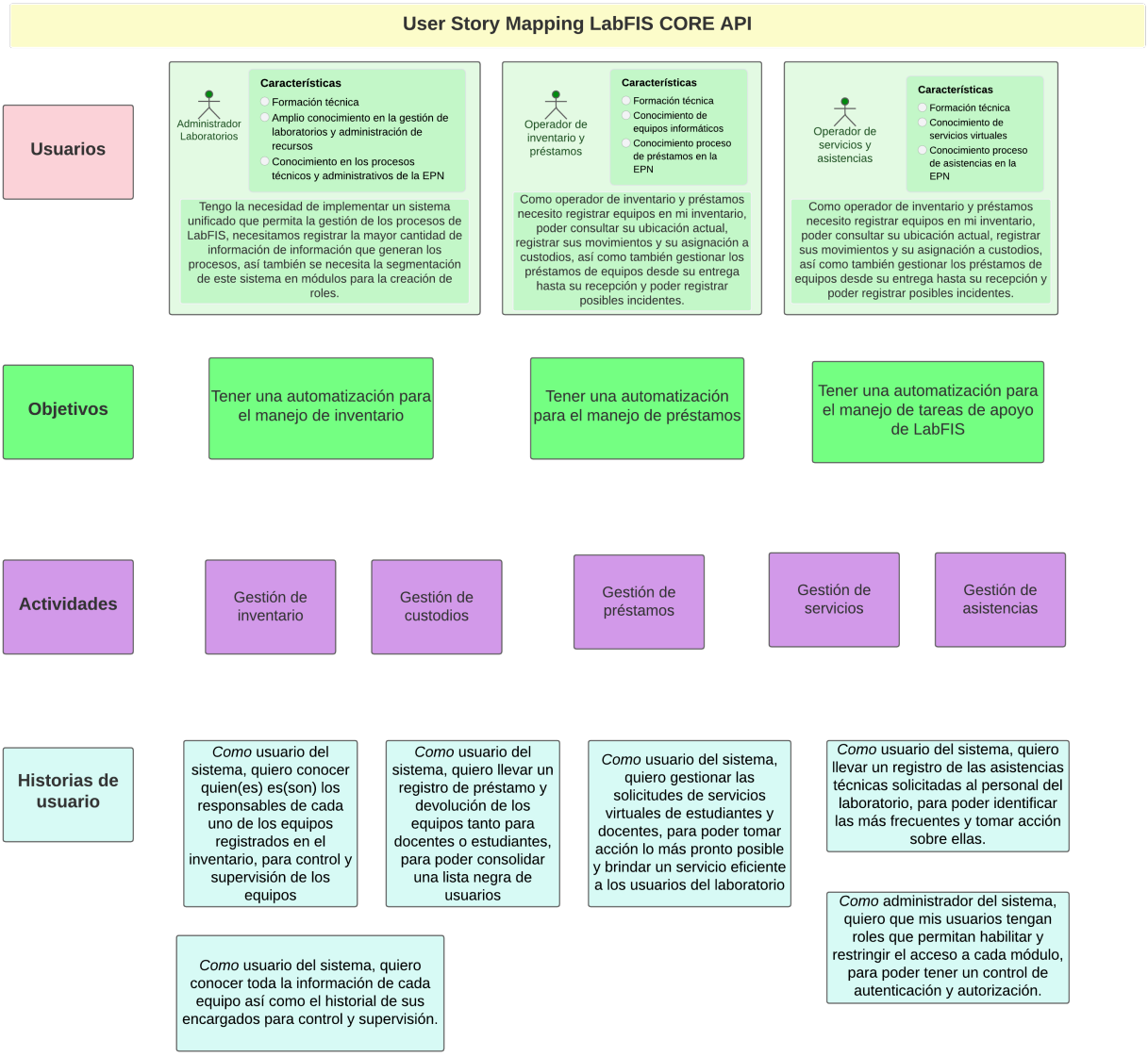
[20] JetBrains, “Infografía del estado del ecosistema del desarrollador en 2021 | JetBrains: Developer Tools for Professionals and Teams,” 2020. Accessed: Aug. 12, 2022. [Online]. Available: https://www.jetbrains.com/es-es/lp/devecosystem-2021/.

[21] S. E. E. Profile, S. E. E. Profile, and S. E. E. Profile, “C o m p a r i s o n o f P r o g r a m m i n g L a n g u a g e s : R e v i e w,” no. July, 2018.

[22] N. Purohit, “A Comparative Study on Open Source Database Management System,” *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 6, no. 4, pp. 792–795, 2018, doi: 10.22214/ijraset.2018.4133.

5 ANEXOS

ANEXO I



ANEXO II

Estándar de Programación y desarrollo

Base de Datos			
Elemento	**Descripción**	**Notación**	**Ejemplo**
Tablas transaccionales	Escrito en mayúsculas, el nombre en singular	TBL_<NOMBRE>	TBL_EQUIPO
Tabla de parámetros	Escrito en mayúsculas, el nombre en singular	TBP_<NOMBRE>	TBP_CARGOS
Procedimientos almacenados	Escrito en minúsculas, verbo en infinitivo con 3 a 4 palabras descriptivas	sp_<verbo>_<palabra>	sp_asignar_equipo_custodio
Atributos de las tablas	Escrito en minúsculas y singular	fld_<atributo>	fld_telefono
Vistas	Escrito en minúsculas, entidades en plural	vista_<entidades>_<caract>	vista_custodios_activos
Funciones	Escrito en minúsculas, inicia con verbo en infinitivo	<verbo>_<atributo>_<palabra>	obtener_ids_problemas_prestamos

Respuestas de procedimientos almacenados	Formato json que puede contener dos elementos, resultado o respuesta	{
		“resultado”: [
		{
		<objeto>
		}
		],
		“respuesta”: {
		“resp”:<string>,
		“infor”: <string>
			{
			“resultado”: [
			{
			<objeto>
			}
			],
			“respuesta”: {
			“resp”:”ok”,
			“infor”: null

		} }	} }
Esquemas	Máximo 6 letras en minúsculas	<abcdef>	Esquema de préstamos de equipos: prequi
API			

Modelo Server	Archivo main del servidor web express	Server.js	Class Server {
			constructor()
			middlewares ()
			routes directories()
			listen()
			}

Capa Routes	Archivos para gestionar las rutas de las peticiones	routes/nombreModulo.routes.js	Imports Router.post('/ruta-ejemplo/', [MiddlewareAuth], controller)

Capa controller	Se encargan como tal de responder a la petición	controllers/nombreModulo.controller.js	Imports
			
			const nombreControlador =
			async(req, res) => {
			try {
			//procesamiento requerido
			//llamada a la función de la capa de negocio
			}
			Catch (error){
			// responder con el error correspondiente
			}
			}

Capa negocio	Se encargan de manipular y validar datos de ser necesario.	negocio/nombreModulo.manipulate.js	Imports

			Async function nombreFunciónManipulate (jsonData) {
			//llamada a la función que maneja los datos
			return respuesta
			}

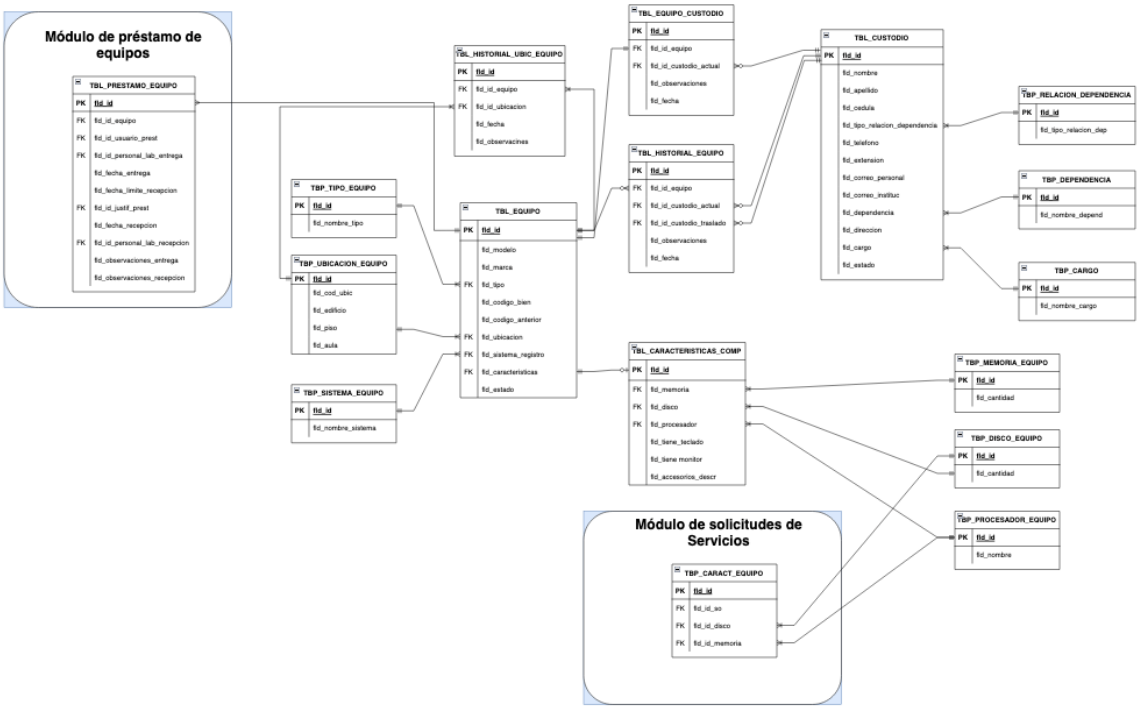
Capa datos	Se encargan de llamar a los procedimientos almacenados de la base de datos (SPs).	datos/nombreModulo.data.js	Imports
			// conectarse a la base
			async function nombreFuncionData(jsonData){
			return response = await pool.query('nombre del SP', datos que requiere el SP)
			}

Tests			

Pruebas unitarias de cada módulo	Archivos que realizan las pruebas unitarias más importantes	testing/nombreModulo.test.js	Imports
			// Definiciones de esquemas básicos para validación
			describe ('Descripción de pruebas', () => {
			test ('Descripción de test', async()=> {
			// implementación de la prueba
			})
			// programar con la misma estructura el resto de tests necesarios.
			})

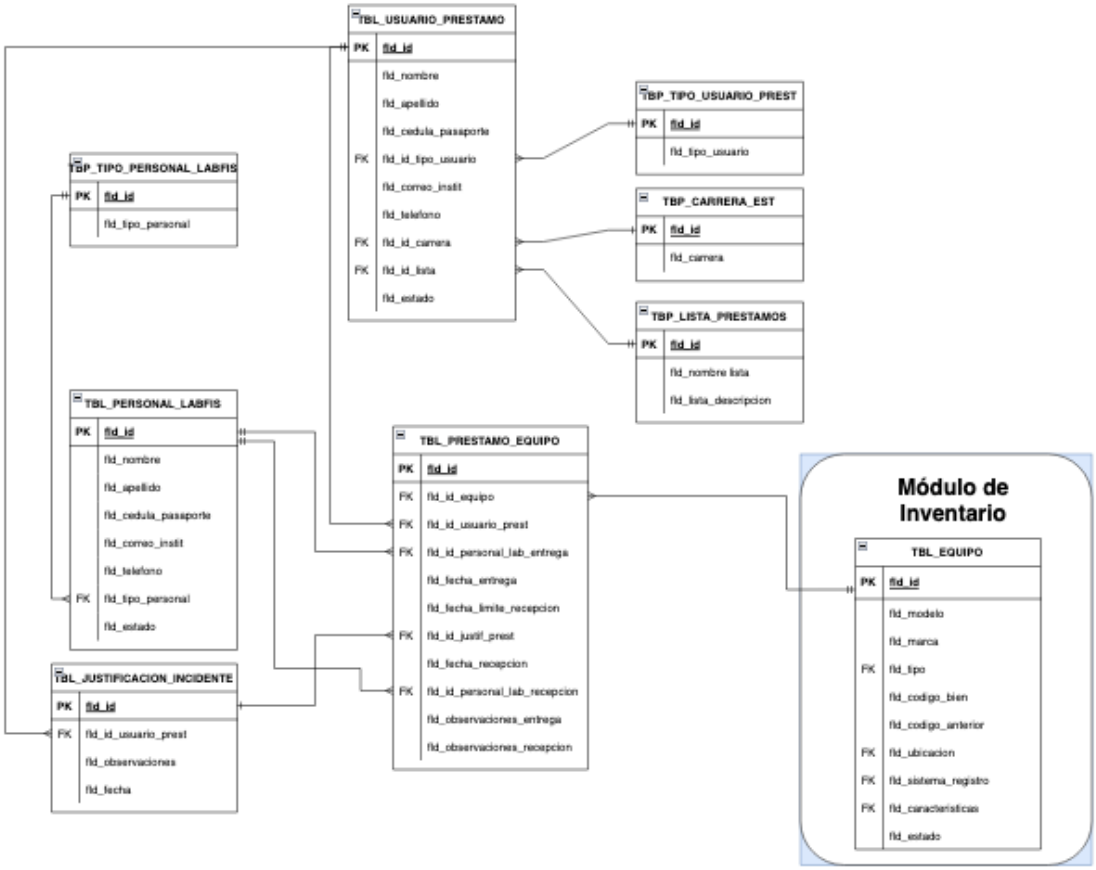
ANEXO III

Diagrama entidad relación de la base de datos del módulo de inventario



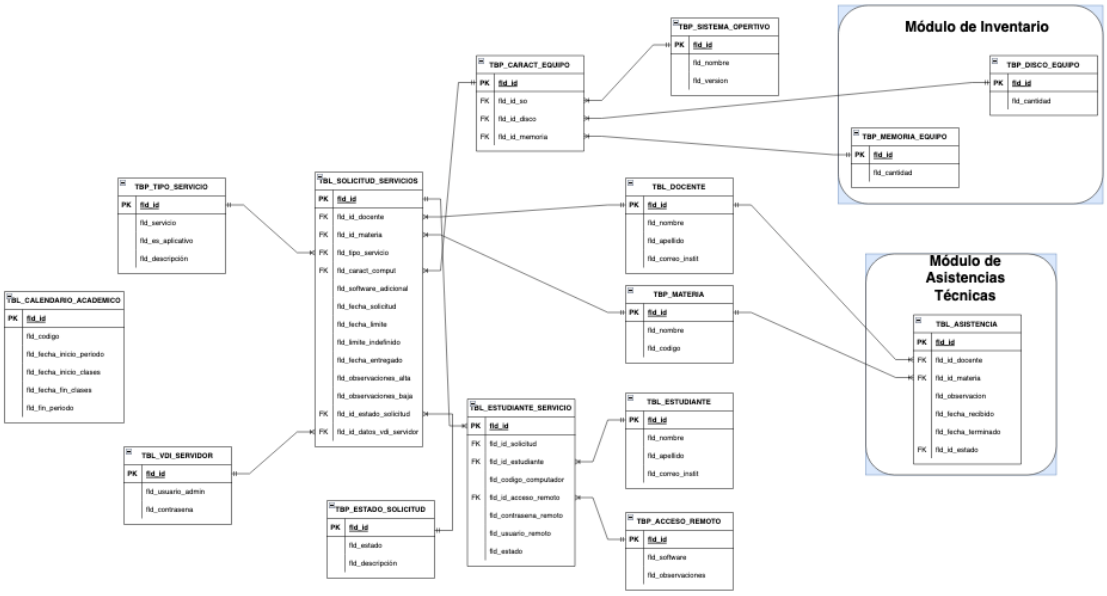
ANEXO IV

Diagrama entidad relación de la base de datos del módulo de préstamo de equipos



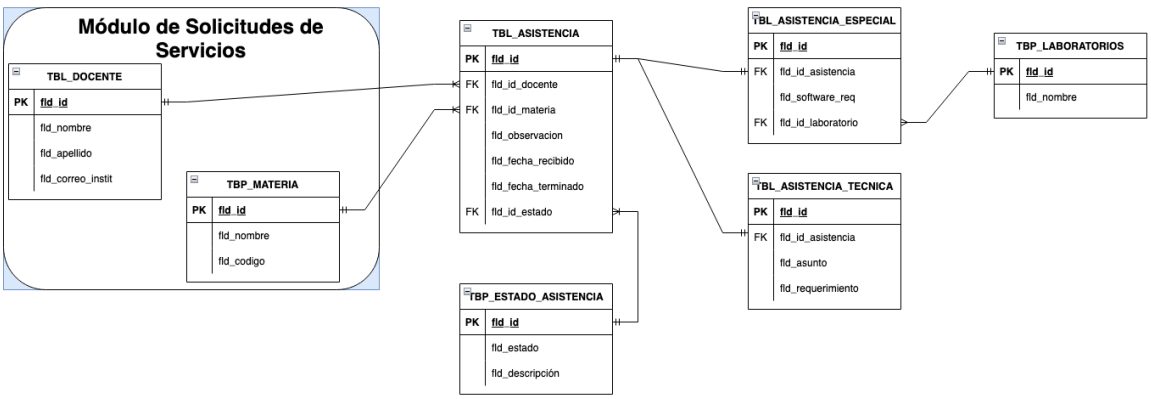
ANEXO V

Diagrama entidad relación de la base de datos del módulo de préstamo servicios



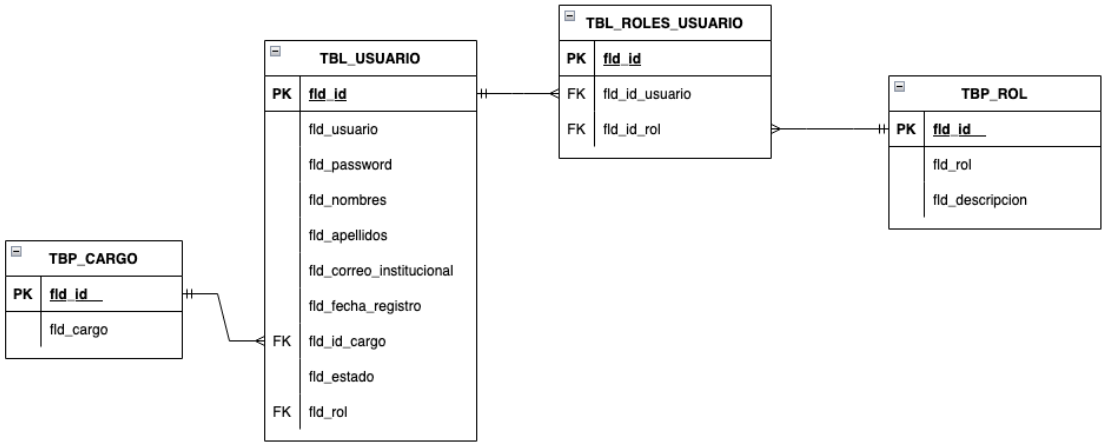
ANEXO VI

Diagrama entidad relación de la base de datos del módulo de asistencias técnicas



ANEXO VII

Diagrama entidad relación de la base de datos del módulo de autorización y autenticación



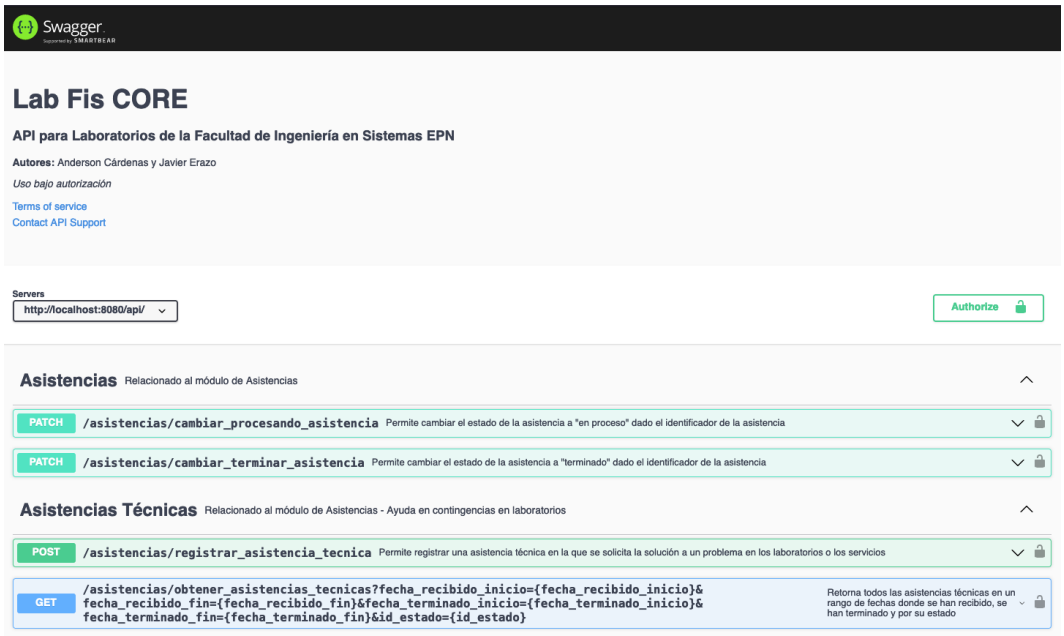
ANEXO VIII

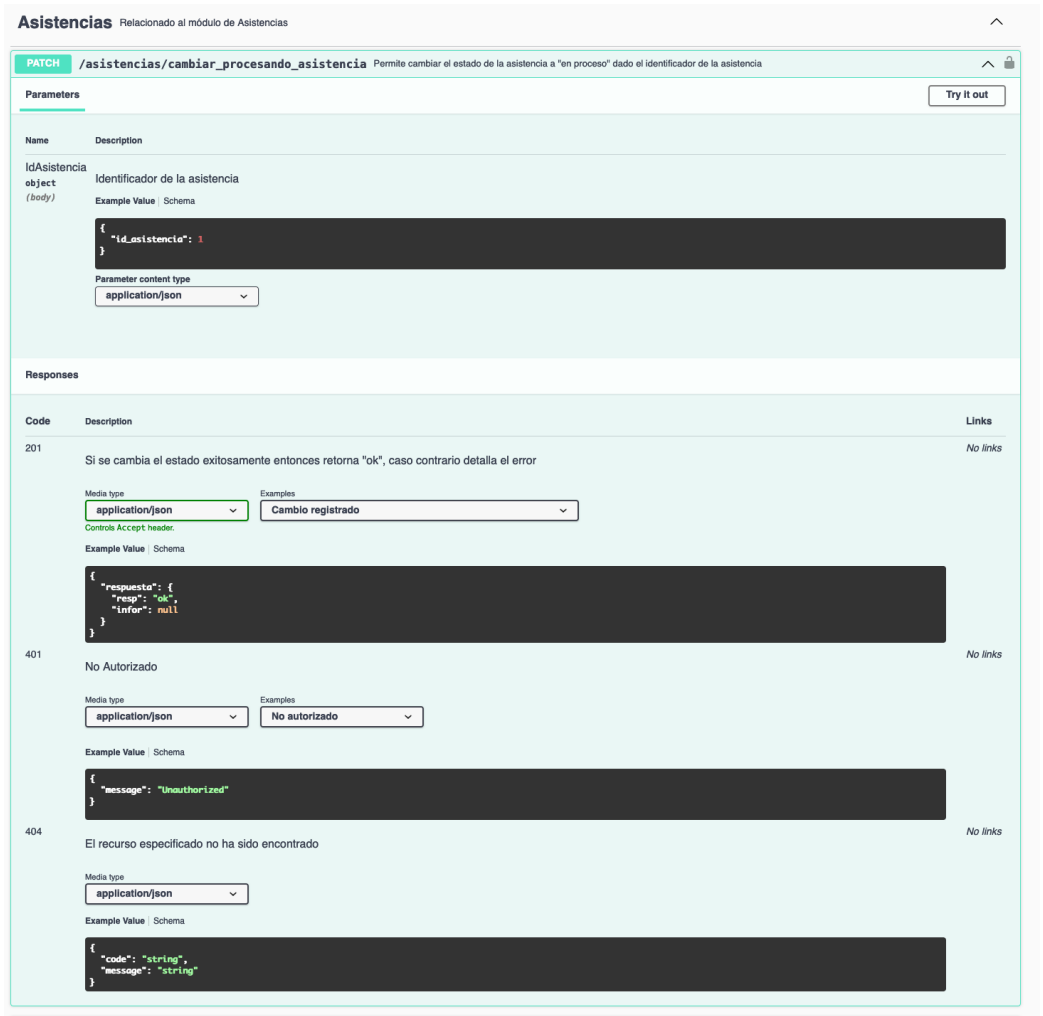
Enlace de repositorio del API

https://gitlab.com/AndersCFR/labfiscore.git

ANEXO IX

Formato de documentación en swagger





ANEXO X

Manual del programador

PostgreSQL

Para la gestión de la base de datos se puede utilizar cualquier gestor, pero se recomienda PostgreSQL por facilidad de uso.

Existen dos bases de datos:

- DB_CORE
- DB_USUARIOS

DB_CORE: Contiene 4 esquemas, uno para cada módulo:

- Invent: relacionado al módulo de inventario
- Prequi: relacionado al préstamo de equipos
- Prserv: relacionado al préstamo de servicios
- Asiste: relacionado a las asistencias técnicas

DB_USUARIOS: contiene solo un esquema llamado auth, el cual gestiona los usuarios que pueden acceder a las operaciones en los diferentes módulos del CORE

Procedimientos almacenados

Todos los procedimientos almacenados reciben y retornan información en formato json. Por otro lado, se utiliza el lenguaje 'plgpsql' para su programación.

Por lo general, se van a tomar 3 parámetros, uno de entrada y dos de retorno, en otros casos donde no se necesite retornar información de la base de datos no se recibirá el parámetro de salida 'resultado', como su muestra en la siguiente porción de código

```
CREATE OR REPLACE PROCEDURE prequi.sp_obtener_usuario_prest_cedula(
        IN json_datos json,
        OUT resultado json,
        OUT respuesta json)
```

Para tomar los elementos del json de entrada se tiene que utilizar la función 'json_array_elements' debido a la estructura de datos (array) que llega desde la API.

```
SELECT j->> 'cedula_pass'
        INTO cedula_pass
        FROM json_array_elements(json_datos) AS j;
```

El retorno de información debe ser siempre insertado en los parámetros de salida para que al terminar el procedimiento almacenado o desacadenemos una excepción los retorne de inmediato, para esto las siguientes dos porciónes de código nos ayudan en la mayoría de las casos.

```
SELECT to_json(campos) INTO respuesta
                        FROM (SELECT Resp, Infor) campos;
```

```
SELECT array_to_json(array_agg(res), FALSE) INTO resultado
        FROM schem.vista_temporal res
```

Vistas

Debido a la normalización de la base de datos se requiere hacer varias combinaciones de tablas, para reducir este código en los procedimientos almacenados se crean vistas para conseguir un conjunto de datos concretos fáciles de acceder. Sin embargo, las vistas siempre se manejarán para uso interno de la base de datos y de los procedimientos almacenados

Funciones

En algunos de los módulos se ha necesitado crear funciones para realizar operaciones específicas que pueden resultar confusas tenerlas en los mismos procedimientos. Para mejorar la legibilidad y el entendimiento del código se utilizan en los procedimientos almacenados, cada una cuenta con una descripción para saber su utilidad.

API con Node JS y Express

Ruta

Las rutas son aquellas en las que podemos especificar las peticiones que recibirá nuestra API además podremos especificar el controlador que atenderá esta petición.

```
router.get('/equipos/obtener_equipos_activos',[inventarioAccess], obtenerEquiposActivos);
```

Controlador

Se encarga de la lógica de recepción de la petición, procesamiento adicional y llamar a la función que entregará el resultado de la API a la petición.

```
const registrarEquipoComputador = async (req, res = response) => {
    try {
        const equipoComputador = new EquipoComputador(req.body.Modelo,
req.body.Marca, req.body.IdTipo,
            req.body.CodigoBien, req.body.CodigoAnterior,
            req.body.IdUbicacion, req.body.IdSistemaRegistro,
            req.body.IdMemoria, req.body.IdDisco,
            req.body.IdProcesador, req.body.TieneTeclado,
            req.body.TieneMonitor, req.body.AccesoriosDescr,
            req.body.Estado
        )
        console.log(equipoComputador)
        console.log(typeof (equipoComputador))

        const jsonData = JSON.stringify(equipoComputador)
        console.log(jsonData)
        const response = await registrarEquipoComputacionalManipulate(jsonData);
        res.status(200).json(response.rows[0]);
    }
    catch (error) {
        res.json({
            msg: 'error en registar equipo computador',
            error: error
        })
```

```
        console.log(error)
    }
}
```

Función de Manipulación y Datos

En conjunto permiten hacer procesamiento adicional de los datos que nos llegan como petición y la capa de datos se encarga de llamar al procedimiento almacenado que obtiene o manipula los datos de la base.

```
// Función de manipulación
async function obtenerEquiposActivosManipulate(){
    const response = await obtenerEquiposActivosData();
    return response
}
// Función de datos
async function obtenerEquiposActivosData() {
    return response = await pool.query(
        'call invent.sp_obtener_equipos_activos($1,$2)',
        convertToEmptyEmpty()
    );
}
```

Protección de una ruta usando un middleware

En caso de que necesitemos limitar el acceso a una ruta únicamente a ciertos roles debemos crear una función que valide el JWT del usuario y sus respectivos roles, con ello podremos determinar si un usuario tiene o no acceso a una ruta, especificamos los middlewares como segundo parámetro en nuestras rutas

```
router.get('/equipos/obtener_equipos_activos',[inventarioAccess], obtenerEquiposActivos);

// inventario
const inventarioAccess = async(req, res, next) => {
    const rolesAdmitidos = ['AdminCore', 'AdminInventario'];
    try {
        const token = req.headers["x-access-token"]
        console.log(req.headers)
```

```
        console.log(token)
        //si no tenemos el token romper comunicación
        if (!token) {
            return res.status(403).json({message: "No acces token provided, you need a x-access-token in HTTP header"})
        };
        const decoded = jwt.verify(token, config.secret);
        const usuario = decoded.usuario;

        const jsonData = JSON.stringify({nombre_usuario:usuario});
        const credentials = await obtenerCredenciales(jsonData);

        // Si no existe este usuario excepción not found
        let userFound = credentials.rows[0].resultado
        if (!userFound) {
            return res.status(400).json({message: "Usuario no registrado"})
        }
        let userRoles = await obtenerRolesUsuario(jsonData);
        userRoles = userRoles.rows[0].resultado;
        let admitedRoles = 0;
        userRoles.forEach(rol => {
            if (rolesAdmitidos.includes(rol.rol))
                admitedRoles+=1;
        });

        if(admitedRoles<1) {
            return res.status(401).json({message: "Unauthorized, no valid user roles"});
        }
        next()
    }catch(error){
        return res.status(401).json({message: "Unauthorized"});
    }
}
```