Portal de documentos Konecta

Pedro Ríos Domínguez

Informe de práctica presentado para optar al título de Ingeniero de Sistemas

Asesores
(Interno) Luis Hernando Silva Flórez, Ingeniero de sistemas
(Externo) Cristian Fabian Castillo López, Ingeniero de sistemas

Universidad de Antioquia
Facultad de Ingeniería
Ingeniería de Sistemas
Medellín, Antioquia, Colombia
2023

Cita		Ríos Domínguez [1]
Referencia Estilo IEEE (2020)	[1]	P. Ríos Domínguez, “Portal de documentos Konecta”, Trabajo de grado profesional, Ingeniería de Sistemas, Universidad de Antioquia, Medellín, Antioquia, Colombia, 2023.

Dedicatoria

A mi familia porque me ha apoyado en todo momento y ha sido pilar fundamental en mi formación profesional y personal.

Agradecimientos

A Dios.
A la empresa MULTIENLACE S.A.S. KONECTA COLOMBIA y a su valioso talento humano por permitirme realizar este proyecto.
A todos los docentes que me acompañaron en mi proceso de formación durante todos estos semestres en la Universidad de Antioquia.

TABLA DE CONTENIDO

LISTA DE FIGURAS

SIGLAS, ACRÓNIMOS Y ABREVIATURAS

API	Application Programming Interfaces
AWS	Amazon Web Services
Bpo	Business process outsourcing
Db	Database
DTO	Data Transfer Object
Jpql	Java Persistence Query Language
MVC	Model View Controller
Sql	Structured query language
SNS	Simple Notification Service

RESUMEN

Konecta es una empresa multinacional enfocada en bpo, contact center y soluciones de software. Tiene presencia en más de 9 países de 3 continentes. Además, cuenta con 15.000 talentos en Colombia que promueven el buen el relacionamiento con clientes mediante soluciones y servicios de calidad.

En la actualidad es recurrente tener que entregar de forma presencial ciertos documentos para la obtención de un bien o servicio, lo cual puede traducirse como un impedimento o dificultad para gran cantidad de usuarios. Por esto y pensando en la comodidad del cliente, la organización Konecta ha emprendido la labor de modernizar este proceso por medio de las nuevas tecnologías, el Portal de documentos de Konecta es una solución cloud desarrollada usando la metodología SCRUM, la cual permite tener a disposición una plataforma web en la cual sus respectivos usuarios gestionen los documentos que les sean requeridos en la adquisición de un producto o servicio.

Palabras clave — **Multitenant, backend, api, documentos.**

ABSTRACT

Konecta is a multinational company focused on bpo, contact center and software solutions. It has presence in more than 9 countries in 3 continents. In addition, it has 15,000 talents in Colombia that promote good customer relations through quality solutions and services.

Nowadays it is recurrent to have to deliver in person certain documents to obtain a good or service, which can translate as an impediment or difficulty for many users. For this reason, and with the customer's convenience in mind, the Konecta organization has undertaken the task of modernizing this process through new technologies, Konecta's document portal is a cloud solution developed using the SCRUM methodology, which allows to have a web platform available in which their respective users can manage the documents that are required in the acquisition of a product or service.

Keywords — **Multitenant, backend, api, documents.**

I. INTRODUCCIÓN

En gran cantidad de procesos de adquisición de un bien o servicio, es necesario que la persona interesada tenga que facilitar distintos documentos que le sean requeridos, para así poder cumplir con el debido proceso. Presentar dichos documentos de manera física puede conllevar a distintas problemáticas, tales como: la obligatoriedad de hacer un desplazamiento de manera física al sitio en donde se deben alojar dichos documentos; también, que la información puede ser revisada por personal que no está autorizado a acceder a su contenido y finalmente, que los datos pueden perderse por el paso del tiempo, accidentes o desastres naturales. El portal de documentos es la solución que permite a los clientes de Konecta, tener la posibilidad de que sus respectivos usuarios puedan adjuntar, en la nube, de manera confiable y segura, los documentos que se les sean requeridos. Para que en una próxima instancia, éstos puedan ser revisados y autorizados por el personal encargado de dicho proceso en la organización.

Para el desarrollo del portal se planteó usar AWS, Java - Spring Boot para la parte del backend, para el frontend el framework ReactJs, y como base de datos MySQL. Todo esto teniendo en cuenta la arquitectura multitenant de la cual se hablará más adelante.

II. OBJETIVOS

A. Objetivo general

Construir el backend del Portal de documentos usando Java - Spring. que permita al usuario hacer toda la gestión de subida de documentos.

B. Objetivos específicos

- Definir la arquitectura base del backend, con base en una arquitectura multitenant que permita el uso de un mismo código para “n” clientes.
- Implementar módulo de inicio de sesión de los usuarios.
- Implementar módulo que permita al usuario observar sus tareas de carga de documento asignadas.
- Implementar módulo que permita al usuario cargar un archivo.

III. MARCO TEÓRICO

Se requirió definir términos relacionados con la solución, para así tener total claridad de los conceptos con los cuales se trabajó:

- **Aplicación Web:** Es la implementación de una lógica de negocio en un servidor web. Las ventajas que ofrecen son principalmente la facilidad de acceso y compatibilidad que estas ofrecen, ya que por medio de un navegador web el usuario puede realizar acciones que dicha aplicación haya dispuesto [1].
- **Modelo MVC:** El modelo MVC [2] es un patrón que busca reducir el esfuerzo de programación en sistemas múltiples y sincronizados por datos, mediante tres capas que conllevan a una separación clara entre los componentes del aplicativo, dichas capas son:
    - **Modelo:** Define las reglas del negocio, y accede a la persistencia de información.
    - **Vista:** Muestra la información del modelo al usuario.
    - **Controlador:** Comunica la vista con el modelo.
- **Java:** Es un lenguaje de programación utilizado para el Backend de diversas soluciones de software. Es fuertemente tipado y de alto nivel.
- **Spring Boot:** Spring Boot es una herramienta que busca simplificar el trabajo con el framework Spring de Java. Ayuda a que la complejidad del desarrollo de nuevos proyectos se vea reducida, ya que dicha herramienta incluye pautas y bibliotecas relevantes para una amplia gama de aplicaciones [3].
- **Base de datos relacional - SQL:** Sistema de gestión de bases de datos relacionales de código abierto, en donde la información persiste en tablas relacionadas, las cuales están conformadas por filas y columnas. Las columnas son las variables que representan a la tabla, mientras que las filas son la información existente en dicha tabla [4].
- **Arquitectura multitenant:** Es una arquitectura de software basada en una única instancia. Una única implementación desde un servidor a disposición de múltiples clientes y usuarios. Esta arquitectura está pensada para que sea posible el manejo de datos de forma separada en caso de que el cliente lo considere necesario [5]. Los datos de una arquitectura multitenant se pueden organizar de las siguientes maneras [6]:

  - **Base de datos compartida:** Es la más usada, y es en la cual se tiene una misma base de datos para todos los clientes, y la diferenciación de estos es a partir del relacionamiento entre tablas.
  - **Base de datos separada:** Es la implementación en donde cada cliente posee su propia base de datos. De esta forma el aplicativo multitenant debe funcionar independientemente del motor de base de datos de cada cliente.
  - **Base de datos híbrida:** Esta implementación es la unión de las anteriores formas de separar los datos, de esta forma los clientes tienen la posibilidad de guardar en su propia base de datos ciertos segmentos de información que les sea de utilidad.
- **API:** Interfaz de un conjunto de bibliotecas o paquetes de software, capacitados para que otro software o programa pueda acceder a estos y ejecutarlos. [7].
- **AWS:** Plataforma en la nube ampliamente adoptada en el mundo del desarrollo de software, ofrece más de 200 servicios de centros de datos tales como AWS S3 y AWS SNS [8].
- **Sonarqube:** Es una plataforma para evaluar código fuente. Es software libre y usa diversas herramientas de análisis estático de código fuente para obtener métricas que pueden ayudar a mejorar la calidad del código de un programa.
- **SCRUM:** Es un proceso o metodología donde se realizan constantemente un conjunto de actividades para trabajar en equipo y obtener mejores resultados en el desarrollo de un proyecto [9].

IV. METODOLOGÍA

La metodología que se usó en el desarrollo del proyecto fue la metodología SCRUM (Ver Fig. 1.), estos fueron los pasos realizados para el desarrollo del trabajo:

- **Contextualización del problema:** El equipo de desarrollo se familiarizó con el proyecto, esto se realizó con una reunión con el equipo anterior encargado y el profesor, ellos explicaron la situación problemática y el funcionamiento del proyecto.
- **Product backlog:** Se realizó una lista con las tareas de los módulos faltantes del proyecto.
- **Sprint Backlog:** Se seleccionaron los objetivos con mayor prioridad y se elaboró una lista con las tareas que se desarrollaron en los sprints.
- **Sprint:** Ciclo de trabajo donde se llevó a cabo el desarrollo de las tareas. Tuvieron una duración de 1 a 2 semanas.
- **Weekly meeting:** Reunión semanal para asesorías y avances del sprint.
- **Product preview presentation:** Reunión de revisión donde se entregó al encargado las tareas completadas durante el sprint, en ésta el tutor realiza las adaptaciones si las cree necesarias. Estas reuniones se realizaron al mismo tiempo que las del Weekly Meeting.

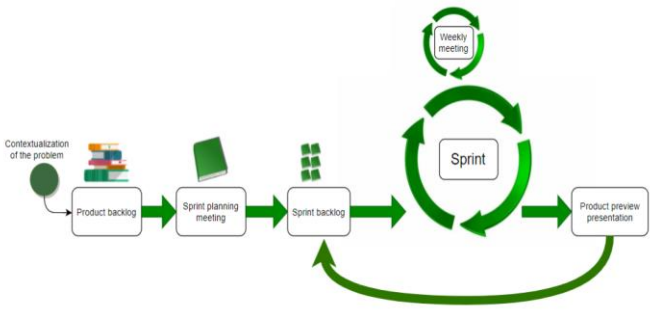

Fig. 1. Ciclo de Scrum

Adicionalmente, para el control de tareas pendientes se utilizó la metodología Kanban gestionada por medio de la plataforma Trello, donde se definieron los siguientes tableros:

- **To do:** Se plantean todas las tareas a realizar a medida que se avanza en el proyecto.
- **Doing:** Se ponen las ideas sobre las cuales se está trabajando.
- **Done:** Tareas realizadas que han sido probadas y están correctamente desarrolladas.

V. RESULTADOS

Se construyó el backend de la solución Portal de documentos, teniendo en cuenta la arquitectura multitenant, en donde si un tenant (cliente de Konecta) decide usar el portal con la característica “Base de datos híbrida” el aplicativo estará en la capacidad de funcionar normalmente, solo es necesario la implementación de unas pequeñas configuraciones parametrizables.

Para la construcción del backend se siguió una metodología ordenada en donde se construyeron las clases del aplicativo de forma que son completamente independientes, desacopladas y reutilizables. Dichas clases se agruparon de forma intuitiva en carpetas (Ver Fig. 2.):

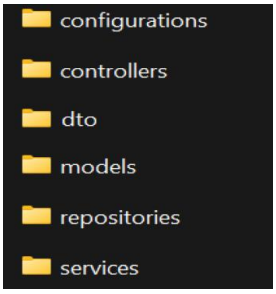

Fig. 2. Estructura general de carpetas en el Backend

A continuación, una breve descripción de cada carpeta:

- **configurations:** En esta carpeta se encuentran todas las clases que tienen que ver con los filtros de seguridad del backend y consumos a recursos externos que deben ser accedidos desde el backend, tales como: AWS S3, AWS SNS y la conexión a las bases de datos de los clientes, en caso de que estos hayan definido tener la configuración multitenant “Base de datos híbrida” (Ver Fig. 3.).

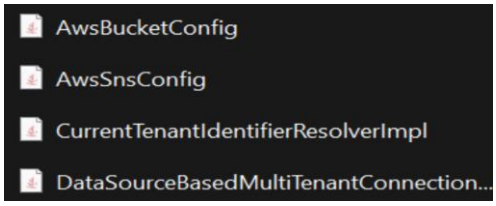

Fig. 3. Ejemplo de clases pertenecientes a la carpeta configurations

- **controllers:** En esta carpeta se encuentran las clases encargadas de la configuración del Servicio REST de la API o Backend (Ver Fig. 4.), cada uno de sus métodos corresponde a un endpoint diferente y en ellos se configuran: el nombre del endpoint, el tipo de método http, el status http, el tipo de respuesta http, la clase DTO que se recibe en la solicitud, el método de la clase service que se va a utilizar para la lógica de negocio y la clase DTO que se entrega en la respuesta, etc.

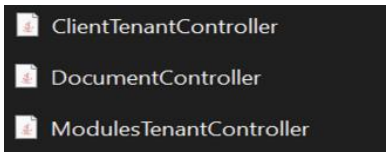

Fig. 4. Ejemplo de clases pertenecientes a la carpeta controllers

- **dto:** En esta carpeta se encuentran las clases utilizadas para la transferencia de datos entre sistemas (Ver Fig. 5.), es decir, que cuando se recibe información por medio de un endpoint, se debe crear una clase DTO que represente esa información y lo mismo para el caso de la respuesta de dicho endpoint.

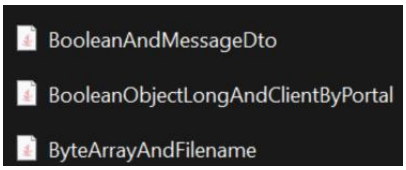

Fig. 5. Ejemplo de clases pertenecientes a la carpeta dto

- **models:** En esta carpeta se encuentran las clases dominio de la aplicación (Ver Fig. 6.), dentro de cada clase se usa la anotación “table” y se indica el nombre de la tabla en la base de datos, también se usa la anotación “column” en los atributos de las clases para hacer referencia a las columnas de la tabla en la base de datos.

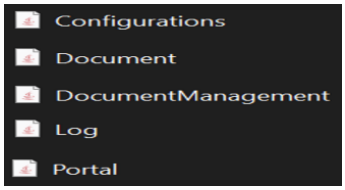

Fig. 6. Ejemplo de clases pertenecientes a la carpeta models

- **repositories:** En esta carpeta se encuentran las clases encargadas de realizar el acceso y persistencia de los datos en las bases de datos (Ver Fig. 7.). Estas son interfaces extendidas de la clase CrudRepository, la cual facilita todo un catálogo de métodos para la persistencia y búsqueda de datos sin tener que realizar queries nativas. Al invocar una clase repositorio, se pueden usar los métodos básicos de CRUD en una base de datos, además, también se posee la flexibilidad de crear métodos java para una búsqueda más avanzada de datos. Si se requieren realizar búsquedas más complejas, la clase también da la flexibilidad de hacer queries nativas en SQL y queries en JPQL.

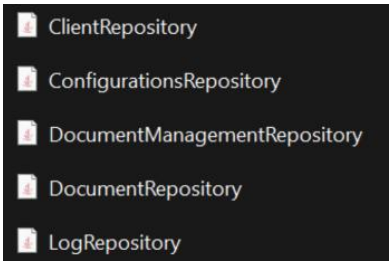

Fig. 7. Ejemplo de clases pertenecientes a la carpeta repositories

- **services:** Allí se encuentran las clases encargadas de realizar la lógica del negocio (Ver Fig. 8.). En estas clases se invocaron los modelos y repositorios necesarios para hacer los respectivos métodos que usan los controladores. Estas clases son las que más flexibilidad poseen, ya que se puede reutilizar cualquier entidad, repositorio y servicio en ellas, además de que la mayoría de sus métodos son independientes y se hicieron de forma de que sean lo más reutilizable posible.

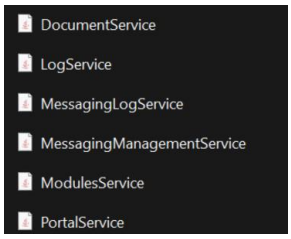

Fig. 8. Ejemplo de clases pertenecientes a la carpeta services

Una vez la estructura del backend fue construida se realizaron las siguientes funcionalidades a partir de todos los objetivos y requerimientos planteados:

- Se implementó una API que permite el inicio de sesión de usuarios al portal (Ver Fig. 9.), en la cual es necesario proporcionarle el tipo de identificación, número de identificación y código aleatorio. Dicho código aleatorio se genera de forma segura y se proporciona al usuario al momento de registrarlo en la base de datos del aplicativo, además este código es de único uso y para poder obtener otro es necesario utilizar un servicio de recuperación de código.

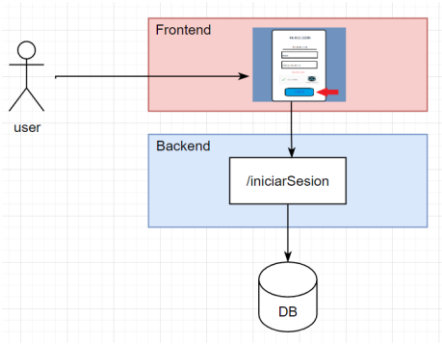

Fig. 9. Diagrama del flujo de inicio de sesión

Cabe aclarar que el registro del usuario en la base de datos y su notificación es responsabilidad de otro aplicativo de la organización.

- Se implementó una API para recuperar o generar un código aleatorio (Ver Fig. 10.), dicho código es enviado al usuario al correo y celular previamente registrados en la base de datos del aplicativo. Se utiliza una herramienta de AWS llamada SNS la cual permite el envío de mensajería de texto y de emails.

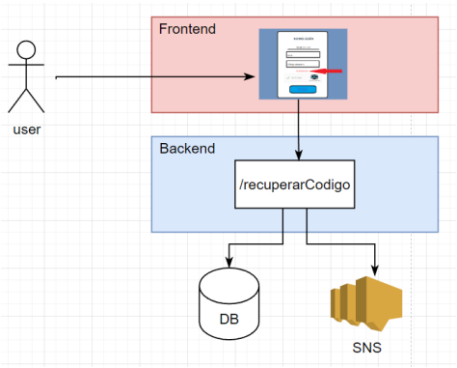

Fig. 10. Diagrama del flujo de recuperación de código aleatorio

- Se implementó una API que permite al usuario recuperar sus tareas de carga de documento asignadas (Ver Fig. 11.), podrá ver el id de la tarea, la fecha de creación, el nombre del producto o servicio que se le está ofreciendo, el estado actual de la tarea y el avance en la carga de archivos requeridos. Esta API es consultada por el Frontend una vez el usuario inicia sesión correctamente.

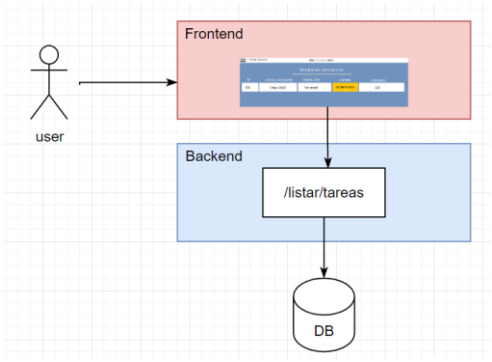

Fig. 11. Diagrama del flujo que devuelve las tareas de un usuario

Cabe mencionar que la creación de las tareas de carga de documentos son responsabilidad de otro aplicativo de la organización.

- Se implementaron los endpoints necesarios para el módulo de subida de documentos, los cuales corresponden a los siguientes procesos:
  - Subida de archivos usando S3 (Ver Fig. 12.).
  - Visualización de archivos alojados en S3.
  - Eliminación de archivos alojados en S3.

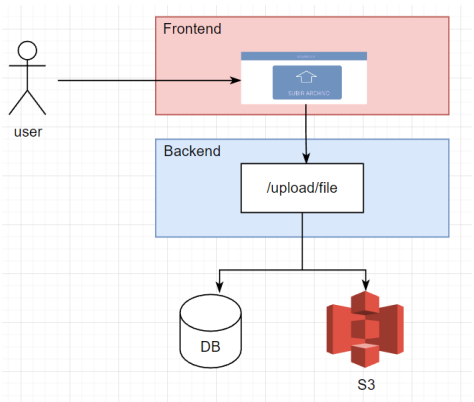

Fig. 12. Diagrama del flujo de carga de documentos

Las funcionalidades de este módulo son usadas en el frontend para que el usuario pueda tener gestión total de todo el procedimiento de subida de archivos en sus respectivas tareas.

- Se implementaron filtros de seguridad por token, los cuales garantizan que un usuario pueda ejecutar únicamente acciones que le correspondan.
- Se usó la herramienta Sonarqube para detectar posibles incidencias de mejoramiento en el código y así poder resolverlas.
- Se apoyó en la construcción y mantenimiento de distintos módulos del frontend.

Como dato importante, la revisión de los documentos que son subidos en el aplicativo por parte de los usuarios corresponde a otro aplicativo de la organización.

VI. TRABAJO A FUTURO

Como trabajo futuro, la empresa evalúa la implementación de distintas opciones que permitan a los usuarios del aplicativo realizar más tareas, las cuales se están refinando y construyendo por parte del equipo de diseño de la organización.

VII. CONCLUSIONES

El desarrollo de este proyecto estuvo lleno de retos y aprendizajes. Es importante resaltar el uso de la metodología SCRUM para llevar a cabo cada sprint, con esto se pudo tener un seguimiento y feedback en todas las etapas de la práctica académica.

La solución fue muy bien pensada debido a que hoy en día se deben buscar formas de facilitarle la vida a los clientes, con el fin de crear un lazo de fidelización.

REFERENCIAS

[1] EcuRed, “Aplicación web” EcuRed. 2019, [En línea]. Disponible en: https://bit.ly/3HCesuc.

[2] Y. Fernández Romero, and Y. Díaz Gonzáles, “Patrón Modelo-Vista-Controlador.” Revista Telem@tica, vol. 11, no. 1, 2012, [En línea]. Disponible en: https://bit.ly/3Ri3s8E.

[3] O. blancarte, “Qué es Spring Boot y su relación con los microservicios” 2018. [En línea]. Disponible en: http://bit.ly/3Rc0CBX.

[4] Gustavo B, “¿Qué es MySQL? Explicación Detallada Para Principiantes,” Hostinger. 2019, [En línea]. Disponible en: http://bit.ly/3jbyIt5.

[5] Nts-solutions, “¿Qué es la arquitectura Salesforce multitenant? ¿Características? ¿Ventajas?” Nts-solutions. 2019, [En línea]. Disponible en: http://bit.ly/3JrTkIi.

[6] D. A. Forero, “Arquitectura multitenant: qué es y por qué es importante,” Platzi. 2017. [En línea]. Disponible en: http://bit.ly/3HjAws0.

[7] C. de C. de Bogotá, “El mundo conectado por las API,” Biblioteca Digital. 2019. [En línea]. Disponible en: http://bit.ly/3Do6J0p.

[8] Amazon, “Informática en la nube con AWS,” Amazon. [En línea]. Disponible en: http://bit.ly/3kGbhIM.

[9] Proyectos Ágiles, “Qué es SCRUM,” Proyectos Ágiles. [En línea]. Disponible en: http://bit.ly/3Y8aJdo.