Aplicación para el proceso de inscripción a la universidad

Yuly Paola Lípez Pinzón
Julian José Castellanos Uribe

Año - 2020

Universidad Antonio Nariño – Bogotá

Especialización en Ingeniería de Software

Trabajo de grado para obtener el título de: Especialista en ingeniería de software

Aplicación para el proceso de inscripción a la universidad

Yuly Paola Lípez Pinzón
Julian José Castellanos Uribe

Asesores

Ing. Dianalin Neme Prada
Ing. Iván Romero Flórez

Año - 2020

Universidad Antonio Nariño – Bogotá

Especialización en Ingeniería de Software

Trabajo de grado para obtener el título de: Especialista en ingeniería de software

Tabla de Contenidos

Lista de tablas

Lista de figuras

Introducción

Cada vez más, la tecnología hace parte de la vida diaria de las personas, de los centros educativos, y las empresas dan a conocer información de interés a través de sitios web.

La educación superior está a la vanguardia en los temas tecnológicos al ofrecer sus programas académicos a través de sitios web con los que buscan motivar a las personas a que se inscriban en dichos programas. Por otra parte, los aspirantes deben buscar en la universidad de su interés las fechas de inscripción y los requisitos de admisión, pero en muchas ocasiones esta información no está completa o actualizada, lo que genera demoras en las inscripciones de los aspirantes, así como un mayor número de solicitudes de soporte al departamento de admisiones para resolver sus dudas.

Desde la especialización en Ingeniería del Software de la Universidad Antonio Nariño se propone una aplicación que guiará paso a paso a los aspirantes en el proceso de inscripción a la universidad de una manera agradable y también permitirá a las universidades ofrecer sus programas de pregrado en plataformas diferentes a las aplicaciones propias.

Formulación y descripción del problema

Según el Sistema Nacional de Información de la Educación Superior (SNIES), los aspirantes inscritos en algún programa académico a corte de junio de 2019 fueron de 2.408.041 personas (SNIES, 2019).

Actualmente las instituciones de educación superior ofrecen en sus sitios web información de sus programas académicos: fechas de inscripción e información de costos y contactos, pero muchas veces esta información está desactualizada o los funcionarios de las instituciones educativas no responden oportunamente las preguntas e inquietudes de los aspirantes que se quieren inscribir en algún programa académico de su interés.

Esta situación ocasiona en la aspirante incertidumbre, preocupación, desatención y aumenta la posibilidad de deserción de la inscripción al programa académico.

Objetivo General

Implementar una aplicación hibrida que acompañe al aspirante en el proceso de inscripción a la universidad.

Objetivos Específicos

- Permitir a los funcionarios de la universidad crear fechas de inscripción, requisitos e información importante para los aspirantes.
- Notificar a los aspirantes el cambio de estado de las diferentes etapas en el proceso de inscripción.
- Permitir el registro de aspirantes y funcionarios dentro de la aplicación.
- Informar del cumplimiento a la protección de datos del aspirante.
- Habilitar a los funcionarios para realizar el seguimiento a los aspirantes inscritos en los diferentes programas de la organización a la cual representan

Marco de Referencia

Estado del arte

Efectuando una búsqueda en internet por las palabras claves “Admisión Universidad”, la mayoría de los resultados corresponden a aplicaciones que ayudan a los aspirantes a estudiar para los exámenes de admisión a entidades de educación superior. Por otra parte, si se busca por el nombre de la universidad, se obtienen resultados de aplicaciones desarrolladas a la medida para su propia comunidad universitaria donde permite consultar notas, horarios e información de eventos que se realizarán en el campus.

Una de las aplicaciones revisadas y que más se acerca al funcionamiento deseado es la aplicación de la universidad EAFIT. Esta aplicación tiene una sección de admisiones y registro académico, donde eligiendo un perfil, ya sea egresado, estudiante (pregrado o postgrado) o aspirante, el usuario puede acceder a las fechas de inscripción de los diferentes programas académicos de pregrado y posgrado ofrecidos por la universidad.

La aplicación permite visualizar las fechas de inscripción a un programa académico seleccionado y permite programar notificaciones para que anuncie al usuario cuando la fecha de inscripción está por caducar. Si un usuario desea seguir con el proceso de inscripción, esta aplicación es solo informativa y no permite visualizar el progreso de la inscripción en las diferentes etapas en el proceso.

Impacto

Esta aplicación nace para los aspirantes y las instituciones de educación superior. Los aspirantes tendrán una herramienta que les permita darle seguimiento a su proceso de inscripción y a las instituciones de educación superior les permitirá agilizar y mejorar este proceso.

Toda la información está disponible en un solo lugar y al alcance del aspirante y de la institución educativa quienes son guiados paso a paso en el proceso de inscripción del aspirante por parte de la aplicación.

Así mismo, las instituciones de educación superior inscritas en la aplicación podrán dar una mayor visibilidad a sus programas académicos ya que serán visualizados por más aspirantes.

Componente de Innovación

Está centrado en el seguimiento que el aspirante puede realizar en todo su proceso de inscripción, desde la selección del programa académico de su interés, hasta la finalización de su inscripción a la institución de educación superior escogida por el aspirante de forma satisfactoria. De igual forma, las universidades inscritas en la aplicación podrán visualizar los aspirantes interesados en cada uno de sus programas académicos por cada etapa del proceso de inscripción.

Marco Teórico

Aplicación Híbrida

Una aplicación Hibrida es una solución y una combinación de una aplicación nativa y una aplicación WEB; una aplicación nativa está inscrita, desarrollada y fundamentada en lenguajes de programación con la tecnología de Javascrypt, CSS y HTML, estos se reúnen y/o combinan para formar una única aplicación con la particularidad que no se ejecuta en el navegador del usuario sino que se ejecuta desde la misma aplicación con un propio navegador integrado el cual es invisible para el usuario; y con el propósito de poderse visualizar en dispositivos móviles y en diferentes navegadores WEB propios de la aplicación. (Griffith, 2020).

Angular

Angular es una plataforma de desarrollo y un marco de trabajo de diseño (Framework) el cual nos permite crear aplicaciones cliente de páginas web simples en una sola página web; utilizando lenguajes de desarrollo HTML y TypeScript. (Angular, 2020).

La primer versión de AngularJS aparece en 2009 por Miško Hevery y originalmente era un servicio de almacenamiento online de archivos tipo JSON; los cuales tenían una única sintaxis para las plantillas; y donde el cobro en rendimiento dependía del peso en megabytes de cada archivo JSON; además del tamaño del paquete con otras bibliotecas y el caótico sistema de bucle hacia que tuviera un bajo rendimiento en la aplicación.

Esta fue la motivación y principal razón que tuvieron los desarrolladores de Google para reescribir el framework que ahora se conoce con el simple nombre de “Angular”; después de tomar el nombre de Angular2, y el cual usa tres pilares fundamentales con en el cual ha sido desarrollado y evolucionado siendo estos: TypeScrypt, RxJS y Zone JS. (Huszárik, 2018)

Este nuevo framework tomo más de 2 años sobrescribirlo y no solo se trató de una nueva versión del framework o una evolución de AngularJS, sino que es un nuevo producto, con sus propios conceptos y técnicas. (CampusMVP, 2020).

Cloud computing

El cloud computing o computación en la nube es un término que se usa con el fin de repartir los recursos de TI (tecnologías de la información) bajo demanda del usuario a través de Internet; pagando por su uso o por un tiempo determinado; sin la necesidad de mantener o comprar servidores físicos y almacenes de datos, se puede tener acceso a servidores tecnológicos, servicios y bases de datos alquilándolos de acuerdo a nuestras necesidades a un proveedor de servicios en la nube de "Cloud Computing" (AWS, 2020).

Amazon Web Services (AWS)

Amazon Web Services es la plataforma de alquiler de servicios y productos de computación en la nube más adoptada y completa en el mundo ofreciendo cerca de 175 servicios integrales de centros de datos a nivel global como lo son; servidores, redes, almacenamiento, correo electrónico, informática remota, desarrollo móvil y de seguridad. (AWS A., 2020).

AWS se divide en tres productos principales los cuales son: EC2 (Servicio de máquina virtual), Glacier (Servicio de almacenamiento de la nube a bajo costo) y S3 sistema de almacenamiento de archivos de Amazon y abarca más de 245 países y territorios a nivel global; estando dividido en 76 zonas de disponibilidad en donde se encuentran sus servidores ´propios y permitiendo a los usuarios establecer límites geográficos. (Page, 2020).

Adoptamos AWS en nuestro proyecto por ser un servicio con mayor funcionalidad en servicios a bajo costo; con grandes clientes, y con una seguridad adaptable y escalable.

Serverless

Serverless es un movimiento que nos permite crear aplicaciones sin servidor y que manejan un tráfico de producción sin ayuda de nada ni de nadie; así dedicamos nuestros esfuerzos en entregar valor al cliente y a nuestros usuarios sin perder el tiempo en servidores ni en nada más. (serverless, 2020).

Serverless es literalmente, -**sin servidor**- y es una arquitectura donde los servidores (físicos o en la nube) dejan de existir para el desarrollador; en cambio el código se ejecuta en "ambientes de ejecución" los cuales son administrados por proveedores como Amazon, Azure, Google, IBM, entre otros. (sPamRucinque, 2017).

Ionic

Ionic Framework es conjunto de herramientas de interfaz de usuario de código abierto para la creación de aplicaciones móviles de alta calidad para sistemas operativos IOS, Android y WEB. Ionic actualmente está construido y desarrollado en lenguajes de programación HTML CSS y JavaScript y hoy en día tiene más del 15% de todas las aplicaciones móviles a nivel global. (DeBeasi, 2020).

Spring Framework

Spring es un framework o marco de trabajo de código abierto originalmente desarrollado en Java y basado en los principios de arquitectura en inyección de dependencias y control de inversión; Su primera aparición se da en el año 2002 por Rod Johnson quien empieza a escribir en su libro las características del framework. Un año después en junio de 2003 se entrega el

primer release del framework con una licencia de Apache 2 versión; actualmente Spring boot se encuentra actualizado en su versión Spring Boot 2.x siendo soportado por Spring 5 y Spring cloud como servicio directamente en la nube como Cloud computing.

Spring framework incluye módulos para el acceso a datos a través de servicios administrando transacciones MVC (Modelo, vista controlador) con la autorización y autenticación de mensajería. (Heartin, 2015).

Metodología

La metodología escogida para el desarrollo de este proyecto es Extreme Programming o (XP). Su principal objetivo es entregar software de calidad basado en potenciar las relaciones interpersonales trabajando en equipo y propiciando un buen clima de trabajo. Está diseñada para equipos pequeños y para proyectos en donde los requerimientos pueden cambiar constantemente. (Agile Alliance, 2020)

Para desarrollar este proyecto, se realizaron sesiones remotas para definir la planeación de actividades, el diseño en la arquitectura que tendría el software, así como la construcción de los diferentes diagramas presentados en este documento distribuyendo las cargas de manera equitativa.

Todo el desarrollo de este proyecto está alineado con la filosofía de la metodología XP, que se basa en los cinco principios de comunicación, simplicidad, retroalimentación, valor y respeto.

Proceso de Software

En el proceso de desarrollo de software se abordarán los requerimientos funcionales y no funcionales, así como los atributos de calidad tenidos en cuenta para sus pruebas. También se explicará el proceso de arquitectura, diseño y construcción de la aplicación.

Requerimientos funcionales

Los requerimientos funcionales son los encargados de definir cómo debe comportarse un sistema, así como cuáles son sus entradas y salidas.

Para esta aplicación, el alcance definido en esta versión permitirá a los aspirantes seleccionar un centro educativo y un programa académico. También permitirá a los funcionarios de los centros educativos configurar las etapas y los periodos académicos requeridos para realizar la inscripción por parte de los aspirantes.

Para una mejor comprensión, los requerimientos se detallan en las siguientes tablas

Tabla 1
Registro en la aplicación

Nombre	Registro en aplicación	RF1
Descripción: Generar un formulario de registro para uso de la aplicación donde se solicitará, datos personales como nombres, apellidos, identificación, centro educativo y programa académico para aspirantes.		
Actores: Aspirantes		
Precondiciones: Ninguna		
Flujo Normal		
Acción del actor	**Respuesta**	
1. El aspirante ingresa los datos solicitados por la aplicación en el formulario de inscripción	2. El Sistema valida los datos ingresados 3. Almacena la información	

	4. Muestra un mensaje, usuario creado exitosamente
Flujo Alternativo	
1. El usuario no ingresa los campos obligatorios	2. El sistema muestra mensaje faltan campos obligatorios
Post condiciones: Ninguna	

Fuente: Elaboración propia

Tabla 2
Crear Organizaciones y programas académicos

Nombre	Crear Organizaciones y programas académicos	RF2
Descripción: El administrador de la aplicación creará la organización y los programas académicos		
Actores: Administrador		
Precondiciones: Ninguna		
Flujo Normal		
Acción del actor	**Respuesta**	
1. El administrador ingresará la información de la organización y los programas académicos pertenecientes a esta	2. El Sistema valida los datos ingresados 3. Almacena la información 4. Muestra un mensaje, organización y programa académico creado exitosamente	
Flujo Alternativo		
5. El usuario no ingresa los campos obligatorios	6. El sistema muestra mensaje faltan campos obligatorios	
Post condiciones: Ninguna		

Fuente: Elaboración propia

Tabla 3
Crear Etapas

Nombre	Crear etapas	RF3
Descripción: El funcionario creara las actividades que se llevan a cabo para la inscripción a los programas académicos		
Actores: Funcionario		
Precondiciones: Ninguna		
Flujo Normal		
Acción del actor	**Respuesta**	
1. El funcionario creará las actividades que se llevaran a cabo en cada periodo de inscripción.	2. El Sistema valida los datos ingresados 3. Almacena la información 4. Muestra un mensaje etapa creada con éxito	
Flujo Alternativo		
5. El usuario no ingresa los campos obligatorios	6. El sistema muestra mensaje faltan campos obligatorios	
Post condiciones: Ninguna		

Fuente: Elaboración propia

Tabla 4
Visualizar seguimiento de aspirante

Nombre	Visualizar seguimiento de aspirante	RF4
Descripción: El funcionario podrá visualizar a cada inspirante inscrito a la organización a la cual representa		
Actores: Funcionario		
Precondiciones: Ninguna		
Flujo Normal		

Acción del actor	**Respuesta**
1. El funcionario entrará a la pestaña de seguimiento y verá una lista de aspirantes inscritos 2. Podrá seleccionar a uno de los aspirantes para visualizar su progreso	3. El Sistema muestra la información del aspirante seleccionado
Flujo Alternativo	
N/A	N/A
Post condiciones: Ninguna	

Fuente: Elaboración propia

Tabla 5
Envío notificación aspirante

Nombre	Envío de notificación por correo electrónico al aspirante	RF5
Descripción: Se enviará un correo electrónico al aspirante cuando haya una modificación por parte del funcionario en su progreso		
Actores: Funcionario		
Precondiciones: Tener en estado diferente a aceptado alguna de las etapas de su progreso		
Flujo Normal		
Acción del actor	**Respuesta**	
1. El funcionario selecciona un aspirante para ver su progreso 2. El funcionario acepta o rechaza la etapa o documento	3. Se envía un correo electrónico al aspirante, anunciando que hubo un cambio en su progreso	
Flujo Alternativo		
N/A	N/A	
Post condiciones: Ninguna		

Fuente: Elaboración propia

Tabla 6
Envío notificación funcionario

Nombre	Envío de notificación por correo electrónico al funcionario	RF6
Descripción: Se enviará un correo electrónico al funcionario cuando el aspirante haya notificado el envío de documentos		
Actores: Aspirante		
Precondiciones: Tener una etapa con documentos a notificar		
Flujo Normal		
Acción del actor	**Respuesta**	
1. El aspirante selecciona el documento que envío previamente al funcionario dando click en el botón entregar.	2. Se le enviará un correo electrónico al funcionario de la organización indicando que un aspirante ha realizado la entrega de un documento	
Flujo Alternativo		
N/A	N/A	
Post condiciones: Ninguna		

Fuente: Elaboración propia

Requerimientos no funcionales

Los requerimientos no funcionales son características y restricciones de la aplicación que ese está desarrollando. Los más relevantes son la usabilidad, seguridad, rendimiento y escalabilidad. A continuación, se detallan los más importantes:

Tabla 7
Requerimiento no funcional Seguridad

Nombre	Seguridad	RNF1
Tipo	Necesario	
Descripción	Las claves deben estar encriptadas en todo el uso de la aplicación	
Criterios de aceptación	Se debe verificar que las claves estén encriptadas en la base de datos	

Fuente: Elaboración propia

Tabla 8
Requerimiento no funcional Usabilidad

Nombre	Usabilidad	RNF2
Tipo	Necesario	
Descripción	La aplicación debe ser intuitiva y fácil de usar Los mensajes deben ser claros La aplicación debe tener diseño que se vea bien tanto en computadores, como en dispositivos móviles	
Criterios de aceptación	Verificación en los diferentes dispositivos y pruebas realizadas	

Fuente: Elaboración propia

Tabla 9
Requerimiento no funcional Escalamiento

Nombre	Escalamiento	RNF3
Tipo	Necesario	
Descripción	Se elige la arquitectura serveless ya que permite que no haya indisponibilidad en la aplicación al ajustarse a las necesidades del momento de su uso	
Criterios de aceptación	Verificación pruebas de carga	

Fuente: Elaboración propia

Diseño y arquitectura

En el proceso de diseño y arquitectura, se describe el caso de uso arquitecturalmente relevante y todos los diagramas usados para su diseño y arquitectura. Estos diagramas permiten de una manera general mostrar cómo está desarrollada la aplicación.

Se ha seleccionado una arquitectura serverless para el desarrollo de la aplicación. Esta arquitectura consta de lo siguiente:

Componente de presentación: Desarrollado en Angular 10 y Ionic 5. Se eligieron estas tecnologías ya que se hace un solo esfuerzo en el desarrollo para las diferentes plataformas (web, IOS, Android) además provee múltiples componentes, elementos que agilizan el desarrollo.

Componente de Desarrollo: Se usó como lenguaje de programación Java 8 ya que es el lenguaje más conocido por el equipo de desarrollo y como framework Spring Boot ya que provee múltiples clases, interfaces, librerías (inyección de dependencias, seguridad, persistencia) que permiten un desarrollo más ágil.

Se usaron dos paradigmas de programación para el desarrollo de esta aplicación: paradigma de programación orientado a objetos y el paradigma de programación reactiva.

Se crearon clases personalizadas para el manejo de errores, de esta manera las excepciones son mucho más fáciles de manejar y entender.

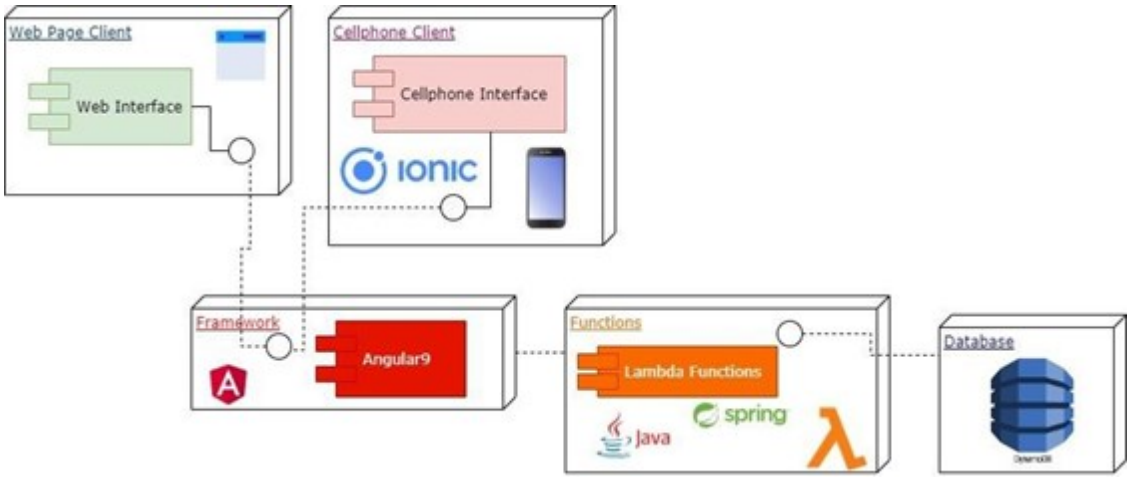
Infraestructura: Se elige el entorno en la nube usando AWS lambdas que contienen toda la lógica de la aplicación, se decide por esta infraestructura ya que las lambdas escalan a la medida de las peticiones. Para la persistencia se usa Dynamodb que es administrada por AWS.

Diagrama de despliegue

A continuación, se relaciona el diagrama de despliegue:

Figura 1

Diagrama de despliegue



Fuente: Elaboración propia

Tabla 10
Caso de uso arquitecturalmente relevante

Caso de Uso	Visualizar seguimiento	
Propósito	A través de la aplicación, el funcionario puede realizar el seguimiento de los aspirantes inscritos a su organización, así como el aspirante ve la información necesaria para su inscripción.	
Actores	Aspirante, funcionario	
Tipo	Primario	
Referencias	Inscripción a programa.	

Precondición	Aspirante debe estar inscrito en una organización y programa académico. Funcionario debe tener aspirantes inscritos en su organización
Postcondición	Para el aspirante visualizar las etapas en curso, finalizadas y que vendrán en el proceso de inscripción Para el funcionario visualizar aspirantes inscritos en la organización

Fuente: Elaboración propia

Tabla 11
Curso Normal Aspirante

Nro.	Ejecutor	Paso o Actividad
1	Aspirante	El aspirante selecciona la opción visualizar seguimiento
2	Sistema	El sistema consulta la organización en la que está inscrito el aspirante y el progreso asociado
3	Sistema	El sistema visualiza las etapas en curso, finalizadas y las que vienen en su proceso de inscripción

Fuente: Elaboración propia

Tabla 12
Curso Normal Funcionario

Nro.	Ejecutor	Paso o Actividad
1	Funcionario	El funcionario selecciona la opción visualizar seguimiento

2	Sistema	El sistema consulta los aspirantes inscritos en la organización a la que pertenece el funcionario
3	Sistema	El sistema visualiza los aspirantes inscritos por programa perteneciente a la organización del funcionario

Fuente: Elaboración propia

Tabla 13
Cursos Alternos Aspirantes

Nro.	Descripción de acciones alternas
2a	Si la organización no ha definido las etapas, saldrá un mensaje informativo diciendo que no se han actualizado las etapas

Fuente: Elaboración propia

Tabla 14
Cursos Alternos Funcionario

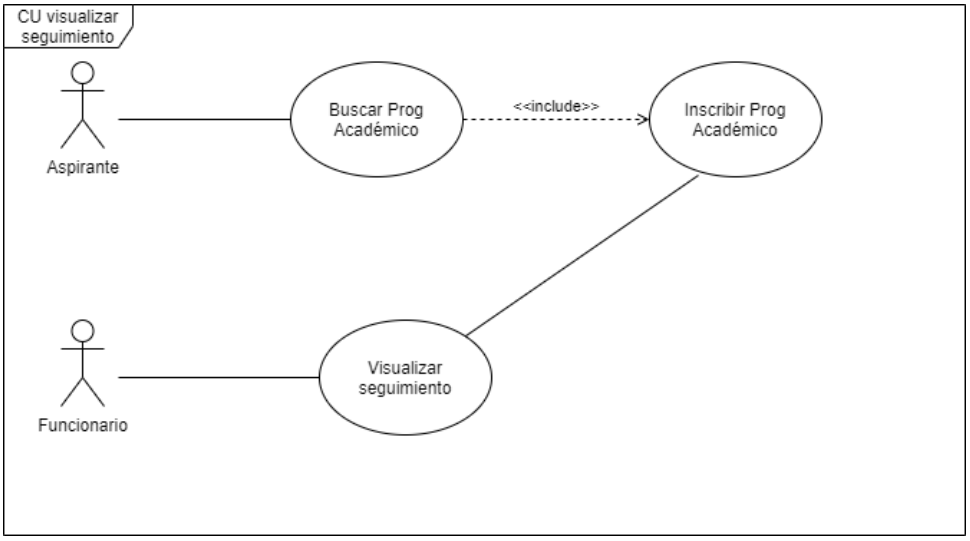
Nro.	Descripción de acciones alternas
2a	Si la organización no tiene aspirantes inscritos saldrá un mensaje informativo diciendo que no hay aspirantes

Fuente: Elaboración propia

Diagrama de caso de uso arquitecturalmente relevante

Figura 2

Diagrama de casos de uso: Visualizar seguimiento



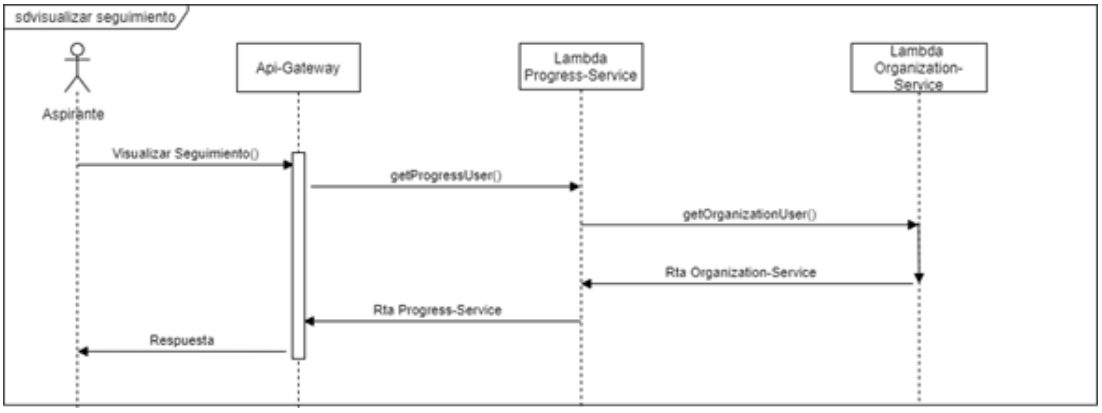
Fuente: Elaboración propia

Diagrama de secuencia

En el diagrama de secuencia se representa la interacción de los objetos que hacen parte del caso arquitecturalmente relevante.

Figura 3

Diagrama de secuencia: Visualizar seguimiento



Fuente: Elaboración propia

Diagrama de clases

En el siguiente diagrama se representan las clases que son relevantes para el desarrollo de la aplicación. En este diseño se observa el principio de responsabilidad única en cada una de las clases y el bajo acoplamiento al usar interfaces para la comunicación entre ellas

Figura 4

Diagrama de clases

Fuente: Elaboración propia

Arquitectura de alto nivel

Construcción

En el proceso de construcción de software, se traducen los requerimientos funcionales y no funcionales a estructura de código fuente que está previamente diseñada, verificada y aprobada por el equipo de desarrollo.

Estas estructuras corresponden a diagramas de casos de uso, despliegue, secuencia, clases y muestran de manera general cómo quedará el software solicitado por el cliente.

A continuación, se mencionan las herramientas utilizadas por el equipo de desarrollo, pruebas realizadas e instalación y configuración de la aplicación

Herramientas de desarrollo

Para el back-end se usaron las siguientes herramientas

- Java 8
- Spring Boot framework
- JSON para transferir datos entre el back-end y el front-end
- IDE IntelliJ y Eclipse
- JWT y Spring Security para el manejo de sesiones y filtros de seguridad

Para el front-end se usaron las siguientes herramientas

- Angular Framework v. 10

- Ionic Framework v. 5
- Visual Studio Code
- NPM como gestor de dependencias de Angular

Bases de datos

- Dynamo DB

Despliegue

- AWS Lambda
- AWS S3
- Infraestructura como código AWS CloudFormation

Se eligen estas tecnologías ya que son las que más se adaptan al diseño de arquitectura de la aplicación y a los conocimientos del equipo de desarrollo.

La construcción de la aplicación se divide en dos fases que son las siguientes:

1- Front-end

Para el desarrollo del front-end se utilizaron los frameworks Angular 10 y Ionic 5. Con estas tecnologías se desarrollaron las diferentes funcionalidades de la aplicación a las cuales tendrán acceso los diferentes tipos de usuarios.

Para el manejo de los datos en la sesión se usa el patrón Redux que hace uso del paradigma de programación reactiva, el cual usa la librería RXJS

2- Back-end

Para la construcción del back-end se utilizó la arquitectura orientada a microservicios. Se eligió esta arquitectura porque permite dividir las diferentes funcionalidades en proyectos independientes que se comunican por medio de APIs. Estos microservicios fueron desarrollados usando el framework Spring Boot.

En cada microservicio se implementaron los paquetes:

- Controller: En este paquete se implementan los métodos encargados de recibir y dar respuesta a las peticiones http
- Repositories: Encargado de recibir los datos que vienen del paquete Services y persistirlos
- Services: Tiene la responsabilidad de recibir las peticiones del paquete Controller, realizar validaciones, invocar otros microservicios y enviar datos al paquete repository.
- Configuration: En este paquete se encuentra toda la configuración de conexión a la base de datos.

El despliegue de estos microservicios se realiza en AWS Lambdas y se hace a través de AWS CloudFormation.

Pruebas

En este capítulo, se abordarán las pruebas realizadas a la aplicación.

El objetivo de estas pruebas es garantizar que se cumplan los objetivos propuestos al inicio del desarrollo. Se realizaron pruebas funcionales comúnmente conocidas como pruebas de caja de negra para validar que se cumplan los requerimientos funcionales, así como pruebas de caja blanca orientadas a pruebas unitarias enfocadas en el front-end.

Pruebas de caja negra

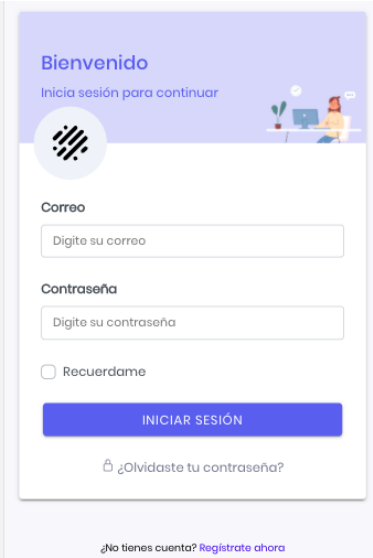
Las pruebas de caja negra se aplicaron a las funcionalidades que hacen parte del caso de uso arquitecturalmente relevante de la aplicación que es visualizar seguimiento. A continuación, se describen las pruebas realizadas.

Creación de usuario aspirante

Inicialmente la aplicación cargará la pantalla de autenticación de usuario

Figura 5

Pantalla de autenticación aplicación

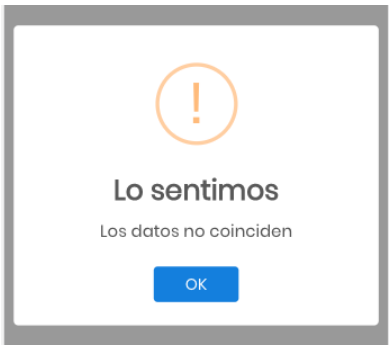


Fuente: Elaboración propia

Si un usuario no está registrado o ingresa mal sus credenciales, se muestra el siguiente mensaje

Figura 6

Mensaje fallo de autenticación

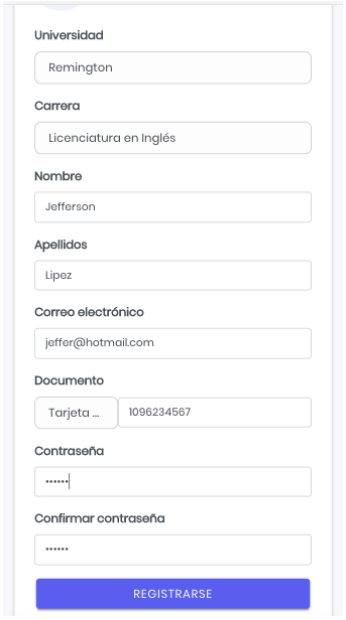


Fuente: Elaboración propia

Para el registro de usuario, se muestra la siguiente pantalla, donde se ingresan los datos solicitados. Se debe aceptar los términos y condiciones de la aplicación presionando el checkbox y luego el botón de registrarse

Figura 7

Pantalla registro aspirante

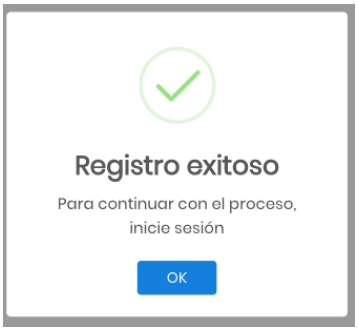


Fuente: Elaboración propia

Si el registro es exitoso, muestra el siguiente mensaje

Figura 8

Mensaje registro exitoso



Fuente: Elaboración propia

Conclusión: La aplicación cumple con los criterios de aceptación de las pruebas de caja negra que no implican el conocimiento del código fuente de la aplicación.

Visualización seguimiento aspirante

Una vez registrado el usuario aspirante, este podrá observar las etapas que se encuentran registradas en el periodo vigente para el centro académico seleccionado.

Figura 9

Pantalla seguimiento aspirante

Fuente: Elaboración propia

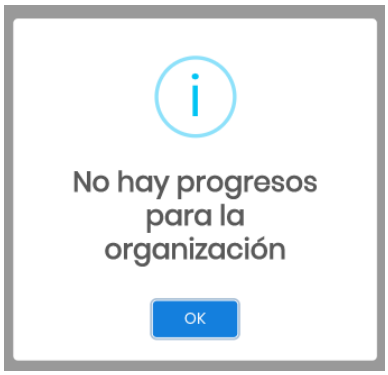
Cada etapa tiene una fecha inicial y una fecha final, así como un estado que indica si la etapa está pendiente, finalizada o en curso.

Creación de periodos

Los usuarios funcionarios pueden crear y gestionar los periodos académicos vigentes. Si la organización o centro educativo no tiene periodos creados mostrará un mensaje y permitirá crearlos.

Figura 10

Mensaje no hay periodos creados



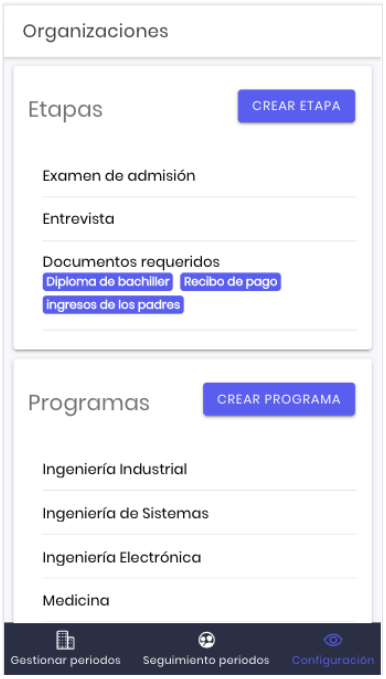
Fuente: Elaboración propia

Antes de la creación del periodo académico, es necesario crear las etapas. Una etapa se define como las actividades que se desarrollarán a lo largo del periodo de inscripción. Esta configuración, así como agregar nuevos programas académicos, se realiza en la pestaña

Configuración

Figura 11

Pantalla Gestión de etapas y programas académicos



Fuente: Elaboración propia

Una vez creadas las etapas, se crea el periodo académico. Las etapas creadas anteriormente se ligan al periodo y se configura las fechas iniciales y finales de cada etapa.

Figura 12

Pantalla Creación de periodo

Fuente: Elaboración propia

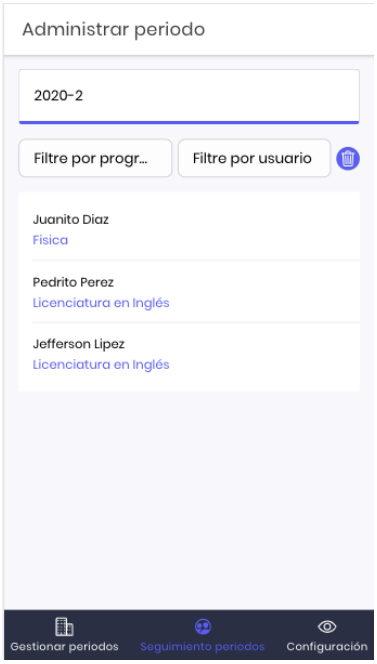
Conclusión: El funcionario puede crear todo el flujo necesario para que los aspirantes puedan registrarse y visualizar su seguimiento. La aplicación cumple con los criterios de aceptación de las pruebas de caja negra.

Visualización de seguimiento por parte del funcionario a los aspirantes

Los funcionarios pueden hacer seguimiento a los aspirantes inscritos en la organización a la cual pertenecen.

Figura 13

Pantalla visualización seguimiento de aspirantes

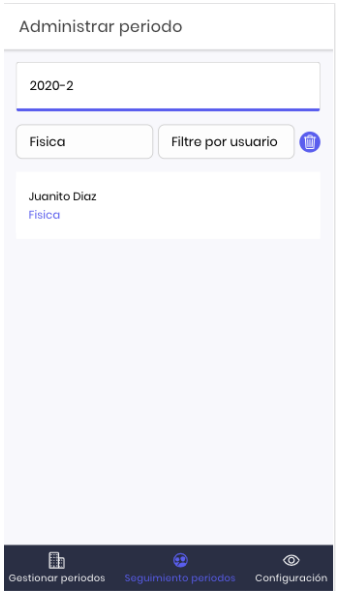


Fuente: Elaboración propia

Pueden visualizar a los aspirantes por programas o por aspirante

Figura 14

Pantalla visualización seguimiento de aspirante por programa académico

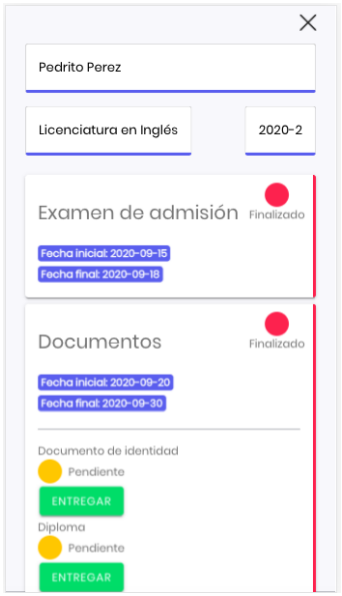


Fuente: Elaboración propia

Una vez aplicados o no los filtros, se puede visualizar el seguimiento de cada aspirante inscrito en la organización.

Figura 15

Pantalla visualización seguimiento por aspirante



Fuente: Elaboración propia

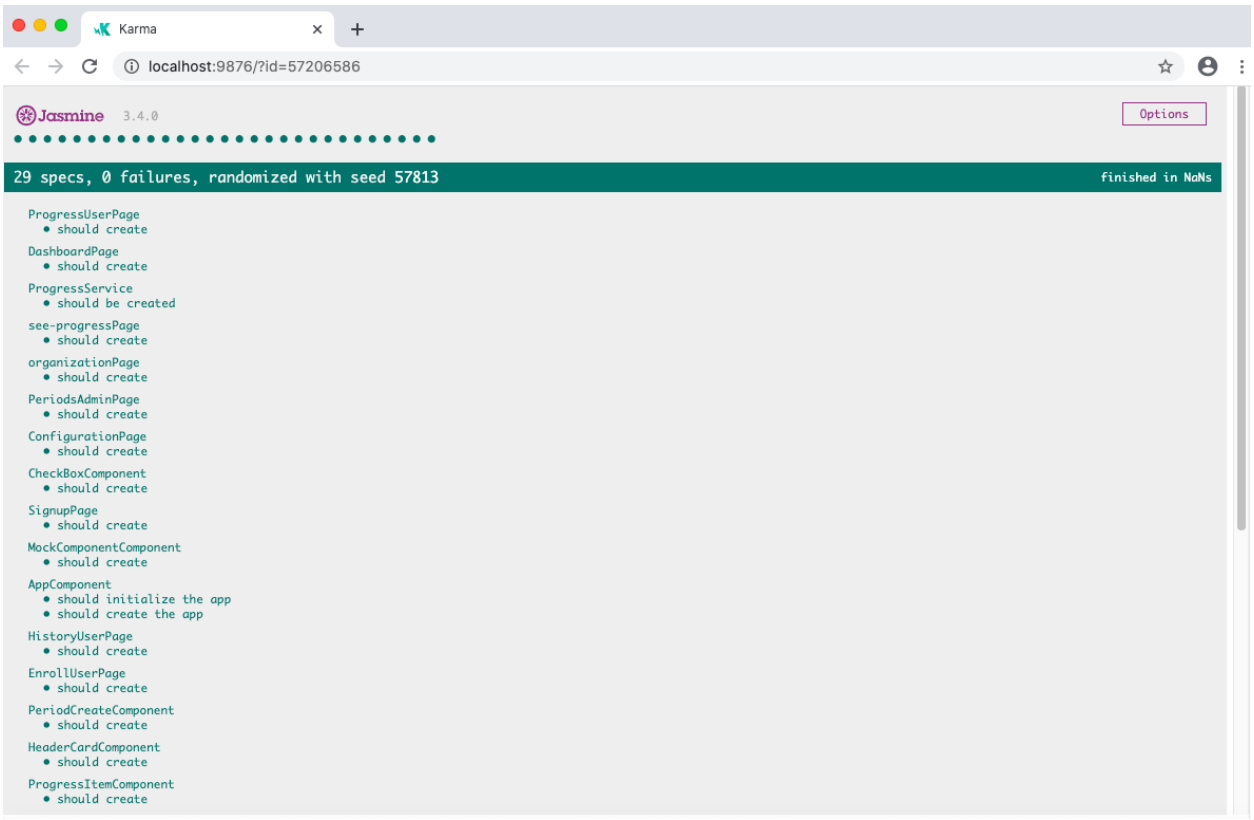
Pruebas de caja blanca

Las pruebas de caja blanca o estructurales están basadas en la evaluación de los detalles procedimentales del código fuente de la aplicación por lo que es necesario conocer la lógica. El objetivo de las pruebas es diseñar casos que se ejecuten al menos una vez, para todas las sentencias del programa y todas las condiciones tanto en su respuesta verdadera como falsa.

Los resultados de pruebas unitarias se ejecutaron a medida que se avanzaba en la codificación de la solución, este tipo de prueba se enfocó en la inicialización de componentes en el front-end utilizando la librería Jasmine

Figura 16

Pruebas unitarias usando Jasmine



Fuente: Elaboración propia

Instalación y configuración

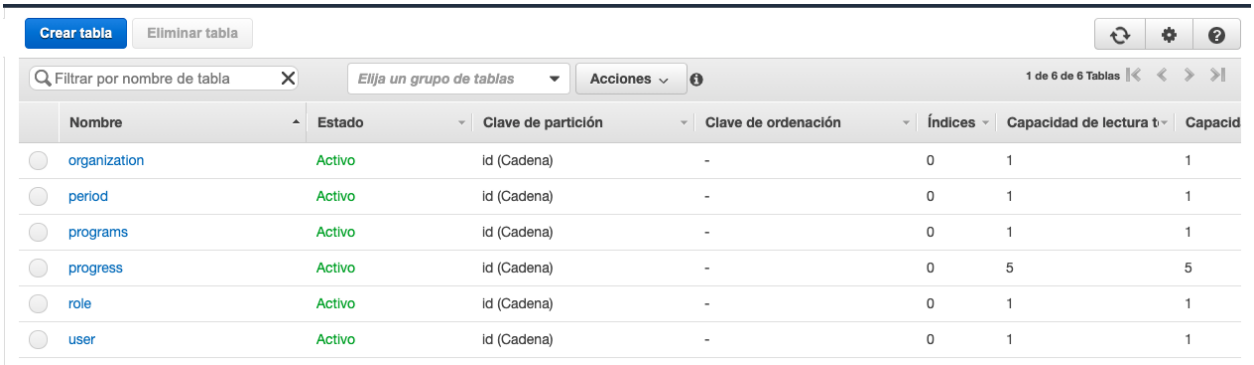
La instalación y configuración se divide en los siguientes pasos

Base de datos

Se decide usar la base de datos DynamoDB. Las tablas se crean a través de la consola de administración de Amazon Web Services.

Figura 17

Consola de administración AWS



Fuente: Elaboración propia

Las funciones lambdas se conectan a la base de datos por medio de las variables de entorno configuradas inicialmente en el archivo de configuración sam.yml y con posibilidad de cambiarlas desde la consola de AWS.

Figura 18

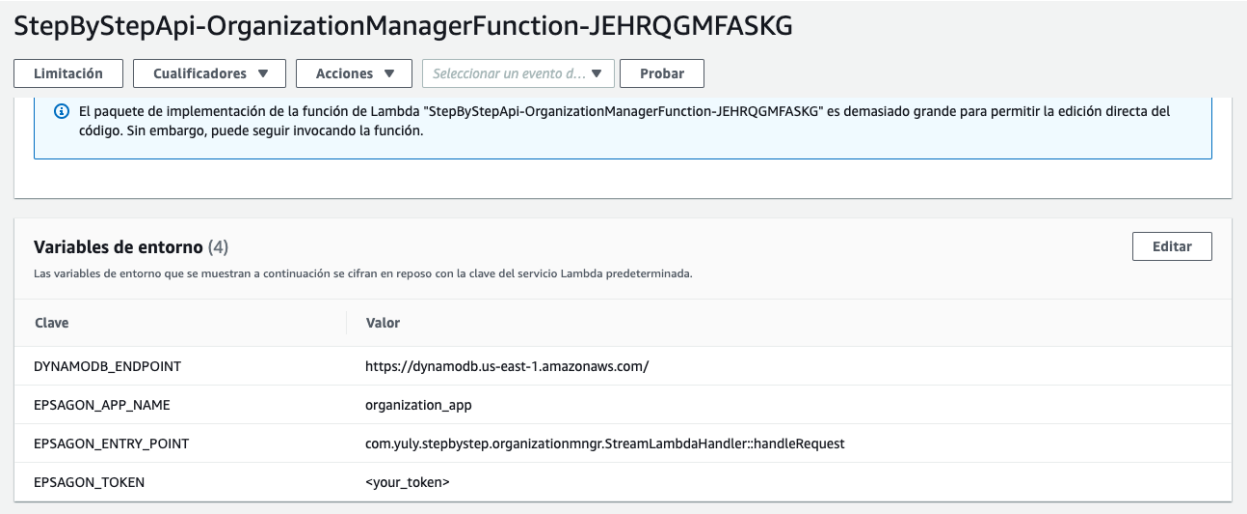
Configuración Base de datos DynamoDB desde sam.yaml

```
UserManagerFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: com.epsagon.EpsagonRequestHandler
    Runtime: java8
    CodeUri: ./user-mngr/target/user-mngr-0.0.1-SNAPSHOT-lambda-package.zip
    MemorySize: 512
    Policies:
      - AWSLambdaBasicExecutionRole
      - AWSLambdaDynamoDBExecutionRole
      - AmazonDynamoDBFullAccess
    Timeout: 30
    Environment:
      Variables:
        EPSAGON_ENTRY_POINT: com.yuly.stepbystep.usermngr.StreamLambdaHandler::handleRequest
        EPSAGON_TOKEN: <your_token>
        EPSAGON_APP_NAME: user_app
        DYNAMODB_ENDPOINT: https://dynamodb.us-east-1.amazonaws.com/
    Events:
      MyApi:
        Type: Api
        Properties:
          Path: /user-mngr/{proxy+}
```

Fuente: Elaboración propia

Figura 19

Configuración Base de datos DynamoDB desde consola de administración AWS



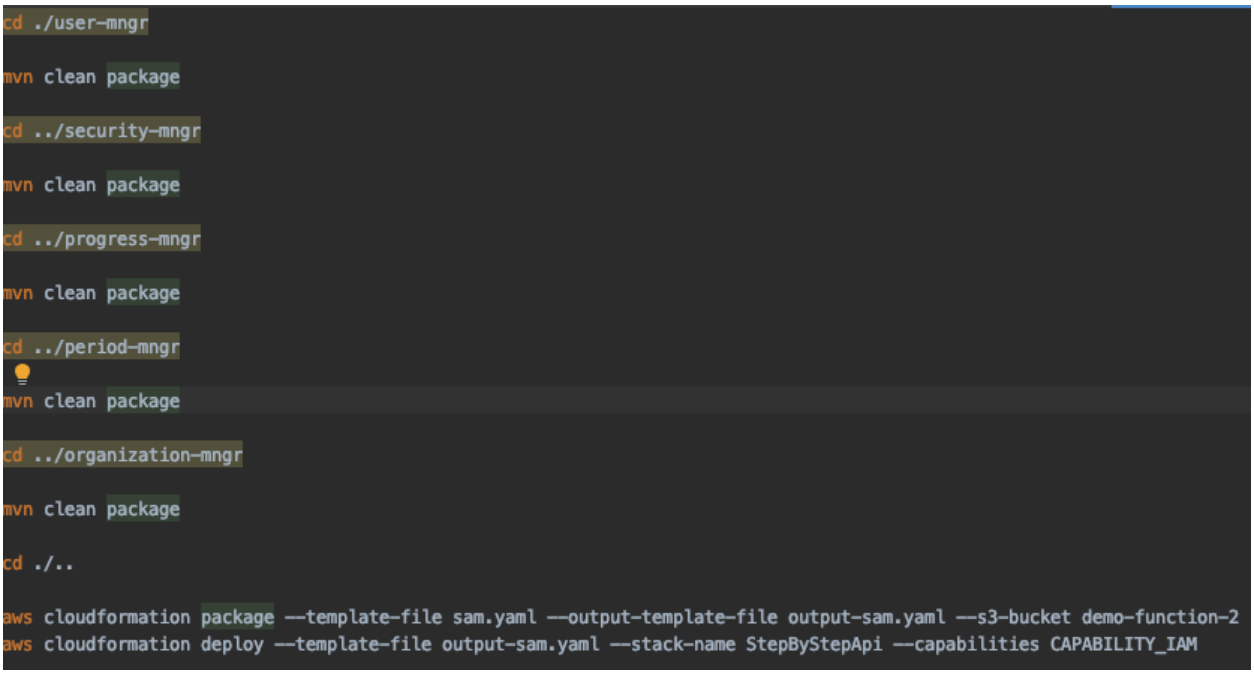
Fuente: Elaboración propia

Configuración y despliegue

Para la compilación y despliegue de las lambdas, se tiene un archivo encargado de esta tarea llamado deploy-aws. Este archivo compila cada manager, luego con el comando “package” de AWS CloudFormation se realizan las acciones escritas en el archivo sam.yaml, generando un archivo output-sam.yaml y con el comando “deploy” se despliegan las lambdas en la nube.

Figura 20

Archivo de configuración deploy-aws



Fuente: Elaboración propia

Archivo sam.yaml

El archivo sam.yml tiene toda la configuración de la infraestructura para el despliegue, y está construido de la siguiente manera:

La primera parte corresponde al API Gateway

Figura 21

Configuración API Gateway

```
ApiResource:
  Type: AWS::Serverless::Api
  Properties:
    StageName: prod
    Cors:
      AllowMethods: "'*'"
      AllowHeaders: "'*'"
      AllowOrigin: "'*'"
```

Fuente: Elaboración propia

La segunda parte corresponde a la configuración de las lambdas

Figura 22

Configuración lambdas

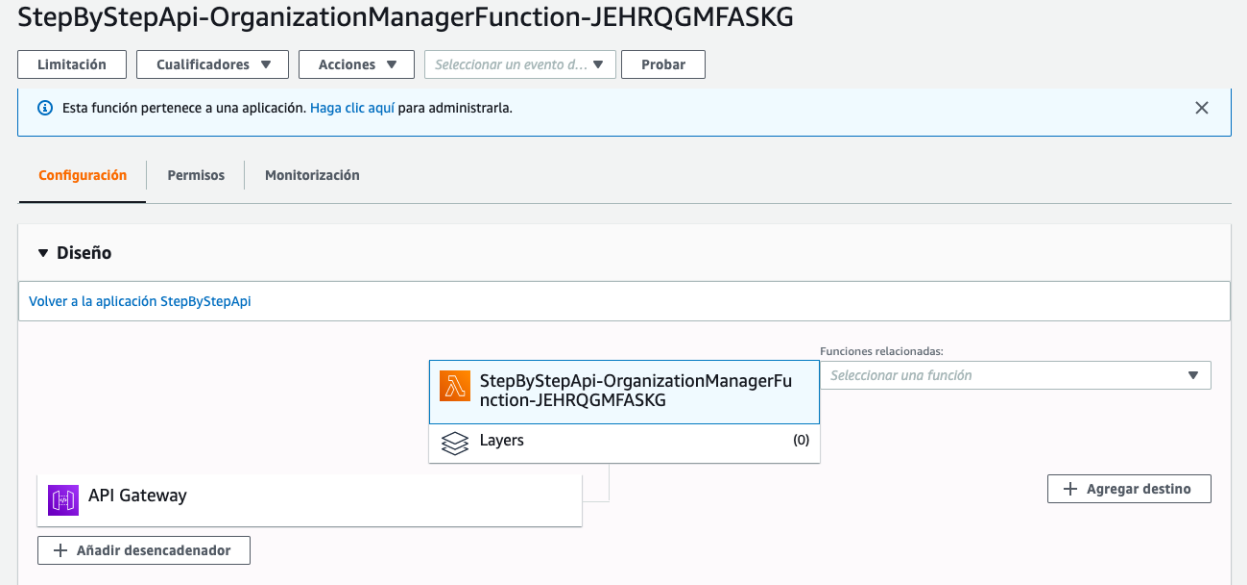
```
ProgressManagerFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: com.epsagon.EpsagonRequestHandler
    Runtime: java8
    CodeUri: ./progress-mngr/target/progress-mngr-0.0.1-SNAPSHOT-lambda-package.zip
    MemorySize: 512
    Policies:
      - AWSLambdaBasicExecutionRole
      - AWSLambdaDynamoDBExecutionRole
      - AmazonDynamoDBFullAccess
    Timeout: 30
    Environment:
      Variables:
        EPSAGON_ENTRY_POINT: com.yuly.stepbystep.progressmngr.StreamLambdaHandler::handleRequest
        EPSAGON_TOKEN: <your_token>
        EPSAGON_APP_NAME: progress_app
        DYNAMODB_ENDPOINT: https://dynamodb.us-east-1.amazonaws.com/
    Events:
      MyApi:
        Type: Api
        Properties:
          Path: /progress-mngr/{proxy+}
          Method: any
          RestApiId: !Ref ApiResource
```

Fuente: Elaboración propia

Una vez ejecutados los comandos anteriormente mencionados, se visualiza la función lambda desplegada en la consola de AWS

Figura 23

Lambda desplegada en la consola AWS



Fuente: Elaboración propia

Conclusiones

- El diseño de la arquitectura se hizo de manera incremental durante el transcurso de la especialización, por lo cual, se logró un diseño detallado, partiendo de una vista de alto nivel hasta llegar al detalle a través de los diagramas descritos en este documento.

- Las funciones como servicio permiten escalar aplicaciones a medida que las peticiones aumenten o disminuyan según sea el caso.

- Desplegar estos microservicios como FaaS permite enfocarse en el desarrollo y el negocio sin preocuparse por su infraestructura y despliegue.
- Aunque el rendimiento de la aplicación es óptimo, los archivos fuentes que se necesitan para desplegar una lambda desarrollada en java son mucho mas pesados que si se usara un lenguaje de programación como python o Node.js los cuales están orientados a FaaS.
- Usar una base de datos NoSQL permite adaptarse mejor a las necesidades del negocio ya que son más flexibles a los cambios.
- El desarrollo de esta aplicación permite que los funcionarios de las instituciones educativas estén enterados del progreso de los aspirantes y así responder de manera oportuna a las solicitudes realizadas
- Usar la arquitectura de microservicios permite realizar cambios en funcionalidades especificas sin afectar a las demás.

Lista de referencias

Alliance, A. (30 de 10 de 2020). *Extreme Programming*. Obtenido de Extreme Programming: https://www.agilealliance.org/glossary/xp/#q=~(infinite~false~filters~(postType~(~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'xp))~searchTerm~'~sort~false~sortDirection~'asc~page~1)

Angular. (08 de 11 de 2020). Introduction to Angular concepts. Obtenido de Introduction to Angular concepts: https://angular.io/guide/architecture
Anonymous, C. P. (01 de 11 de 2020). Security of JSON Web Tokens (JWT). Obtenido de Security of JSON Web Tokens (JWT): https://cyberpolygon.com/materials/security-of-json-web-tokens-jwt/#:~:text=For%20JWT%20signature%20symmetric%20encryption,token%20has%20not%20been%20signed.

AWS. (03 de 11 de 2020). AWS Shield Managed DDoS protection. Obtenido de AWS Shield Managed DDoS protection: https://aws.amazon.com/shield/?nc1=h_ls&whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc

AWS. (01 de 11 de 2020). What is cloud computing? Obtenido de What is cloud computing?: https://aws.amazon.com/what-is-cloud-computing/?nc1=h_ls

AWS, A. (30 de 10 de 2020). *Cloud computing with AWS*. Obtenido de Cloud computing with AWS: https://aws.amazon.com/what-is-aws/?nc1=h_ls

CampusMVP. (22 de 02 de 2020). *Las 5 principales ventajas de usar Angular para crear aplicaciones web.* Obtenido de Las 5 principales ventajas de usar Angular para crear aplicaciones web: https://www.campusmvp.es/recursos/post/las-5-principales-ventajas-de-usar-angular-para-crear-aplicaciones-web.aspx

DeBeasi, L. (15 de 10 de 2020). *Announcing Ionic Vue*. Obtenido de Announcing Ionic Vue: https://ionicframework.com/blog/announcing-ionic-vue/

Griffith, C. (10 de 10 de 2020). *What is Hybrid App Development?* Obtenido de What is Hybrid App Development?: https://ionicframework.com/resources/articles/what-is-hybrid-app-development

Heartin. (02 de 06 de 2015). *INTRODUCTION AND HISTORY OF THE SPRING FRAMEWORK*. Obtenido de INTRODUCTION AND HISTORY OF THE SPRING FRAMEWORK: https://www.javajee.com/introduction-and-history-of-the-spring-framework

Huszárik, M. (13 de 04 de 2018). *AngularJS to Angular - a brief history with some tips to get started!* Obtenido de AngularJS to Angular - a brief history with some tips to get started!:

https://blog.risingstack.com/angularjs-to-angular-history-and-tips-to-get-started/#:~:text=The%20first%20version%20of%20the,day%20front%2Dend%20application%20development.&text=Due%20to%20its%20unpredictable%20change,began%20using%20really%20powerful%20lib

Page, V. (01 de 05 de 2020). *What Is Amazon Web Services and Why Is It so Successful?* Obtenido de What Is Amazon Web Services and Why Is It so Successful?: https://www.investopedia.com/articles/investing/011316/what-amazon-web-services-and-why-it-so-successful.asp

serverless. (10 de 10 de 2020). *Manifesto*. Obtenido de Manifesto: https://www.serverless.com/learn/manifesto/

SNIES. (30 de 12 de 2019). *Sistema Nacional de Información de la Educación Superior*. Obtenido de Sistema Nacional de Información de la Educación Superior: https://snies.mineducacion.gov.co/portal/

SNIES. (22 de 02 de 2020). *Sistema Nacional de Información de la Educación Superior*. Obtenido de Información poblacional, Estadisticas históricas de la educación superior en Colombia: https://hecaa.mineducacion.gov.co/consultaspublicas/content/poblacional/index.jsf

sPamRucinque. (04 de 06 de 2017). *Qué es eso de serverless?* Obtenido de Qué es eso de serverless?: https://medium.com/@PamRucinque/qu%C3%A9-es-eso-de-serverless-f4f6c8949b87