Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

IMPLEMENTACIÓN DEL MÓDULO DE SOLICITUDES DE BECAS PARA LA SECCIÓN SOCIOECONÓMICA DE LA DIRECCIÓN GENERAL DE DOCENCIA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Glen Abra-ham Calel Robledo
Asesorado por MSc. Marco Tulio Aldana Prillwitz

Guatemala, junio de 2023

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

IMPLEMENTACIÓN DEL MÓDULO DE SOLICITUDES DE BECAS PARA LA SECCIÓN SOCIOECONÓMICA DE LA DIRECCIÓN GENERAL DE DOCENCIA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

GLEN ABRA-HAM CALEL ROBLEDO
ASESORADO POR MSC. MARCO TULIO ALDANA PRILLWITZ

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, JUNIO DE 2023

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANA	Inga. Aurelia Anabela Cordova Estrada
VOCAL I	Ing. José Francisco Gómez Rivera
VOCAL II	Ing. Mario Renato Escobedo Martinez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Kevin Vladimir Armando Cruz Lorente
VOCAL V	Br. Fernando José Paz González
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

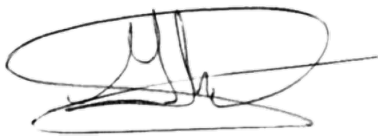
DECANA	Inga. Aurelia Anabela Cordova Estrada
EXAMINADORA	Inga. Floriza Felipa Avila Pesquera de Medinilla
EXAMINADOR	Ing. Sergio Leonel Gómez Bravo
EXAMINADOR	Ing. Carlos Alfredo Azurdia Morales
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

IMPLEMENTACIÓN DEL MÓDULO DE SOLICITUDES DE BECAS PARA LA SECCIÓN SOCIOECONÓMICA DE LA DIRECCIÓN GENERAL DE DOCENCIA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha 29 de enero de 2022.



Glen Abra-ham Calel Robledo

Guatemala, 20 de febrero de 2023

Ingeniero
Oscar Argueta Hernández
Director de la Unidad de EPS
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas
Universidad de San Carlos de Guatemala

Estimado Ingeniero Argueta:

Por medio de la presente doy por finalizado satisfactoriamente el informe de EPS titulado: "IMPLEMENTACIÓN DEL MÓDULO DE SOLICITUDES DE BECAS PARA LA SECCIÓN SOCIOECONÓMICA DE LA DIRECCIÓN GENERAL DE DOCENCIA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA".

Dicho informe fue elaborado por el estudiante: GLEN ABRA-HAM CALEL ROBLEDO quien se identifica con registro académico 201314642 y código único de identificación 3716001110101, de la carrera en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala.

Sin otro particular me despido, Atentamente.

Marco Tulio Aldana Prillwitz
Master in Bussiness Intelligence
and Data Analytics
Colegio de Humanidades 30747

MSc. Marco Tulio Aldana Prillwitz
Asesor de EPS
Escuela de Ingeniería en Ciencias y Sistemas

Universidad de San Carlos de Guatemala

Facultad de Ingeniería
Unidad de EPS

Guatemala, 01 de marzo de 2023.
REF.EPS.DOC.117.03.2023.

Ing. Oscar Argueta Hernández
Director Unidad de EPS
Facultad de Ingeniería
Presente

Estimado Ingeniero Argueta Hernández:

Por este medio atentamente le informo que como Supervisora de la Práctica del Ejercicio Profesional Supervisado, (E.P.S) del estudiante universitario de la Carrera de Ingeniería en Ciencias y Sistemas, **Glen Abra-ham Calel Robledo, Registro Académico 201314642 y CUI 3716 00111 0101** procedí a revisar el informe final, cuyo título es **IMPLEMENTACIÓN DEL MÓDULO DE SOLICITUDES DE BECAS PARA LA SECCIÓN SOCIOECONÓMICA DE LA DIRECCIÓN GENERAL DE DOCENCIA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA.**

En tal virtud, **LO DOY POR APROBADO**, solicitándole darle el trámite respectivo.

Sin otro particular, me es grato suscribirme.

Atentamente,

"Id y Enseñad a Todos"

Inga. Floriza Felipa Ávila Pesquera de Medinilla
Supervisora de EPS
Área de Ingeniería en Ciencias y Sistemas

FFAPdM/RA

Universidad de San Carlos de Guatemala

Facultad de Ingeniería
Unidad de EPS

Guatemala, 01 de marzo de 2023.
REF.EPS.D.76.03.2023.

Ing. Carlos Gustavo Alonzo
Director Escuela de Ingeniería Ciencias y Sistemas
Facultad de Ingeniería
Presente

Estimado Ingeniero Alonzo:

Por este medio atentamente le envío el informe final correspondiente a la práctica del Ejercicio Profesional Supervisado, (E.P.S) titulado **IMPLEMENTACIÓN DEL MÓDULO DE SOLICITUDES DE BECAS PARA LA SECCIÓN SOCIOECONÓMICA DE LA DIRECCIÓN GENERAL DE DOCENCIA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA,** que fue desarrollado por el estudiante universitario **Glen Abra-ham Calel Robledo, Registro Académico 201314642 y CUI 3716 00111 0101** quien fue debidamente asesorado por el Lic. Marco Tulio Aldana Prillwitz y supervisado por la Inga. Floriza Felipa Ávila Pesquera de Medinilla.

Por lo que habiendo cumplido con los objetivos y requisitos de ley del referido trabajo y existiendo la aprobación del mismo por parte del Asesor y la Supervisora de EPS, en mi calidad de Director apruebo su contenido solicitándole darle el trámite respectivo.

Sin otro particular, me es grato suscribirme.

Atentamente,
"Id y Enseñad a Todos"

Universidad de San Carlos de Guatemala
DIRECCIÓN
Unidad de Prácticas de Ingeniería y EPS
Facultad de Ingeniería

Ing. Oscar Argueta Hernández
Director Unidad de EPS

/ra

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala 28 de marzo de 2023

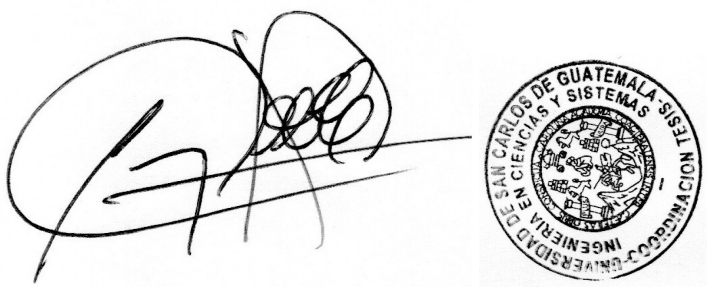
Ingeniero
Carlos Gustavo Alonzo
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Alonzo:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación-EPS del estudiante **GLEN ABRA-HAM CALEL ROBLEDO** carné **201314642** y CUI **3716 00111 0101,** titulado: **"IMPLEMENTACIÓN DEL MÓDULO DE SOLICITUDES DE BECAS PARA LA SECCIÓN SOCIOECONÓMICA DE LA DIRECCIÓN GENERAL DE DOCENCIA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA"** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,



Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

LNG.DIRECTOR.124.EICCSS.2023

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del Asesor, el visto bueno del Coordinador de área y la aprobación del área de lingüística del trabajo de graduación titulado: **IMPLEMENTACIÓN DEL MÓDULO DE SOLICITUDES DE BECAS PARA LA SECCIÓN SOCIOECONÓMICA DE LA DIRECCIÓN GENERAL DE DOCENCIA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA**, presentado por: **Glen Abra-ham Calel Robledo** , procedo con el Aval del mismo, ya que cumple con los requisitos normados por la Facultad de Ingeniería.

"ID Y ENSEÑAD A TODOS"

Ing. Carlos Gustavo Alonzo
Director
Escuela de Ingeniería en Ciencias y Sistemas

MSc. Ing. Carlos Gustavo Alonzo
Director
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, mayo de 2023

Decanato
Facultad de Ingeniería
24189101- 24189102
secretariadecanato@ingenieria.usac.edu.gt

LNG.DECANATO.OI.484.2023

La Decana de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al Trabajo de Graduación titulado: **IMPLEMENTACIÓN DEL MÓDULO DE SOLICITUDES DE BECAS PARA LA SECCIÓN SOCIOECONÓMICA DE LA DIRECCIÓN GENERAL DE DOCENCIA DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA,** presentado por: **Glen Abra-ham Calel Robledo,** después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANA
FACULTAD DE INGENIERÍA

Inga. Aurelia Anabela Cordova Estrada
Decana

Guatemala, junio de 2023

AACE/gaoc

ACTO QUE DEDICO A:

Dios	Por otorgarme vida, salud y la fuerza necesaria para alcanzar este logro, por haber estado en los momentos más complicados y darme la fuerza para seguir adelante, al Padre, Hijo y Espíritu Santo sea la honra, la gloria y el honor.
Mis padres	Juan Calel y Florili Roblero, por todo su amor, apoyo, comprensión, esfuerzo y sacrificio, por haberme enseñado a enfrentar el temor y nunca rendirme, por ser un gran ejemplo de superación, mi mayor inspiración y motivo para seguir adelante.
Mi hermana	Nathaly Calel, por su apoyo, ayuda y ejemplo de disciplina en el desarrollo de mi carrera.
Mis tíos	Por su apoyo, ánimo y aportes que me brindaron en el desarrollo de mi carrera.
Mis primos	Por su amistad, afecto y ánimo que me acompañaron durante la carrera.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por ser la casa de estudios que me dio la oportunidad para formarme como profesional y por todas las emociones que me permitió vivir durante el transcurso de la carrera.
Facultad de Ingeniería	Por todas las herramientas, experiencias y conocimientos que me brindaron durante todos estos años.
Sección Socioeconómica	Por brindarme la oportunidad y la confianza en la realización de este proyecto.
Mis asesores	MSc. Marco Aldana, Lic. Rodrigo Mendizabal e Inga. Gladys Aceituno por compartir sus conocimientos, proporcionarme su ayuda y apoyo en la realización de este proyecto.
Mis amigos	A todos mis compañeros de universidad, especialmente a Henry López, Diego Fuentes y Byron Cermeño, por su compañía y ayuda, por haberme compartido sus conocimientos y brindarme su apoyo en el transcurso de la carrera.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
Gb	Gigabyte
GHz	Gigahercio
Mb	Megabyte
TB	Terabyte

GLOSARIO

API	*Application Program Interfaz***.** Es un conjunto de instrucciones que permite el intercambio de datos entre aplicaciones.
Bug	Es un error o una falla en el software causado por instrucciones incorrectas o faltantes.
Compilación	Proceso donde el sistema informatico convierte un conjunto de instrucciones de alto nivel en un leguaje de máquina.
Dashboard	Es un tablero que le permite visualizar al usuario un conjunto de datos.
Endpoint	URL que permite la recepción o transmisión de información que ha sido solicitada por una API.
Función Almacenada	Conjunto de instrucciones escritas en SQL que se encargan de ejecutar una tarea o una operación y debe retornar un valor.
Hardware	Componentes físicos que conforman un sistema electrónico o un sistema informático.

Licencia MIT	*Massachusetts Institute of Technology***.** Licencia que permite al usuario final del software usar, copiar, modificar, distribuir, entre otros.
MySQL	Sistema de gestión de bases de datos relacionales de código abierto.
Procedimiento Almacenado	Conjunto de instrucciones escritas en SQL que se encargan de ejecutar una tarea o una operación dentro de la base de datos.
Retroalimentación	Respuesta del receptor que indica si se está comprendiendo el mensaje o hay que hacer una modificación.
TypeScript	Lenguaje de programación orientado a objetos de código abierto.
URL	*Uniform Resource Locator*. Recurso único que se utiliza para localizar un recurso en la Web.

RESUMEN

Los estudiantes de la Universidad de San Carlos de Guatemala que afrontan dificultades económicas tienen la oportunidad de acudir a una entidad que es la encargada de proporcionarle una solución a su situación. Esta entidad es la Sección Socioeconómica que pertenece a la División de Bienestar Estudiantil de la Universidad de San Carlos de Guatemala, que se encarga de realizar el estudio e investigación de la situación socioeconómica de la población universitaria.

Esta entidad es la más antigua de la División de Bienestar Estudiantil, la cual almacena un gran archivo físico que contiene todas las solicitudes hechas por los estudiantes. Ante la acumulación de expedientes físicos en la actualidad resulta una actividad compleja y laboriosa llevar a cabo la gestión de solicitudes de becas. Uno de los problemas principales es la recepción de documentación por parte de los estudiantes, teniendo en cuenta que el proceso que requiere una solicitud de beca es extenso.

A causa de la situación descrita anteriormente, se ha desarrollado una aplicación web que facilite y agilice la recepción de la documentación solicitada a los estudiantes que estén en el proceso de solicitud de beca. Además, permite que cada solicitud tenga una gestión de documentación ordenada, donde los trabajadores sociales proceden a cargar el informe domiciliar, informe de diagnóstico y expediente completo, de tal manera que la secretaría de comisión procesa a realizar la revisión de los informes y el cálculo de promedio para posteriormente enviar los cuadros de propuesta a jefatura. Esta aplicación web

es totalmente independiente de la plataforma SIIF y está desarrollada con el objetivo de que sea un módulo externo.

Por último, los trabajadores sociales son capacitados y el sistema se está ejecutando en el servidor que proporcionó el Departamento de Procesamiento de Datos de la Universidad de San Carlos de Guatemala, para que todos los usuarios involucrados en el proceso de solicitud de beca puedan hacer uso de este módulo.

OBJETIVOS

General

Crear un módulo a la Sección Socioeconómica (SSE) de la Dirección General de Docencia División de Bienestar Estudiantil de la Universidad de San Carlos de Guatemala, para agilizar el proceso de solicitudes de becas.

Específicos

1. Proveer un módulo para que el estudiante pueda realizar la carga de todos los documentos requeridos dependiendo su tipo de solicitud de beca y conozca el estado de los documentos enviados (aprobado o denegado).

2. Crear un módulo para que el trabajador de la sección socioeconómica pueda verificar la documentación requerida en cada solicitud y este pueda aceptarla o denegarla.

3. Proporcionar un módulo para que la Secretaría de Comisión pueda recibir los informes de visita domiciliar y expedientes completos de las solicitudes de becas.

4. Realizar manuales de usuario del sistema para el personal de la Sección Socioeconómica.

5. Establecer un módulo de reportes para visualizar métricas específicas y sus resultados, por medio de número de solicitud o registro académico, en un determinado tiempo.

6. Implementar un gestor de archivos que permita a la secretaria de comisión y secretaria de convenios enviar, visualizar y eliminar.

7. Integración del módulo en el servidor propio de la Sección Socioeconómica que proporcionó el Departamento de Procesamiento de Datos.

.

INTRODUCCIÓN

Los estudiantes que desean aplicar para obtener una beca en la Universidad de San Carlos desconocen el proceso que implica, un proceso que requiere de una gran cantidad de documentos, firmas, entrevistas y varias revisiones, por ende, este requiere de una gran cantidad de tiempo, que además se ve afectado debido a que el sistema para realizar dicha solicitud no cubre todas las necesidades de la Sección Socioeconómica División de Bienestar Estudiantil de la Universidad de San Carlos.

Ante esta situación se ha propuesto un módulo para que el proceso actual de las solicitudes de becas se agilice y solvente la mayoría de las necesidades que actualmente requiere la Sección Socioeconómica. El módulo tendrá la capacidad para recepción de documentos por parte de los estudiantes, visualización del estado de la solicitud de beca, los trabajadores sociales podrán realizar la carga de sus informes de visita domiciliar y diagnóstico, y la secretaría de comisión podrá realizar la gestión de todos estos documentos.

El módulo tiene un cronograma establecido para su eficiente desarrollo. Al finalizar el proyecto, cada usuario involucrado tendrá la capacidad para utilizarlo, incluyendo al Departamento de Procesamientos de Datos, de tal manera que el proceso de la solicitud de beca beneficie a todos los involucrados.

1. FASE DE INVESTIGACIÓN

1.1. Antecedentes de la empresa

La Sección Socioeconómica pertenece a la División de Bienestar Estudiantil de la Universidad de San Carlos de Guatemala, es la entidad encargada de realizar el estudio e investigación de la situación económica de la población universitaria. Además, es la dependencia técnica-administrativa y de servicio que tiene la responsabilidad de apoyar al estudiante universitario de escasos recursos y alto rendimiento académico. Esta entidad es la única responsable en la administración y ejecución del programa de becas de la Universidad de San Carlos de Guatemala.

1.1.1. Reseña Histórica

La Sección Socioeconómica es la entidad más antigua de la División de Bienestar Estudiantil, esta se creó debido a la aprobación de un servicio de protección económico social del estudiante universitario que se realizó en un Congreso Centroamericano en el año 1948. Para el siguiente año se funda la primera unidad de servicio de bienestar, el cual consistía en un comedor universitario que suministraba los tres tiempos de comida. A partir de este evento surgieron nuevas necesidades a las que fueron suplidas por medio de la creación de otros servicios, en el año 1950, tales como residencia estudiantil, servicio médico, exención de pago de matrícula, entre otros.

El día 10 de octubre de 1959 se creó la Sección Socioeconómica, siendo esta una de las tres secciones del Departamento de Bienestar Estudiantil, con el

objetivo de apoyar a los estudiantes desde un enfoque médico, psicopedagógico y socioeconómico. Otro evento histórico ocurre el día 23 de julio de 1975 en el Acta No. 16-75, donde el Consejo Superior Universitario aprueba el Proyecto de Reglamento de Bienestar Universitario, el cual lo describe como el organismo técnico para la detección de problemas socioeconómicos, de orientación vocacional y de salud en general que esté afectando a la población universitaria de la Universidad de San Carlos de Guatemala.

Para el año 1981 conforme al Acuerdo de Rectoría No. 699-81, se le asigna la categoría de División de Bienestar Estudiantil, como dependencia de la Dirección General de Administración, la cual estaba conformada por la Sección Socioeconómica, Sección de Orientación Vocacional y la Unidad de Salud. En este año, la Sección Socioeconómica manejaba la exclusividad de la administración del Programa de Becas Préstamo de Pregrado.

En el año 1999 el Consejo Superior Universitario por medio del Acta No. 21-99, autorizó la creación de la Dirección General de Docencia, de esta manera fue ubicada la División de Bienestar Estudiantil y las demás entidades dependientes de esta. En este año, la Sección Socioeconómica seguía con el programa de Becas Préstamo y se llevaron a cabo las becas no reembolsables debido a las donaciones de instituciones extranjeras, empresas privadas y personas particulares.

El Consejo Superior Universitario en la sesión del 25 de enero del 2012, aprobó el Reglamento de Becas de Pregrado para estudiantes de la Universidad de San Carlos de Guatemala, en el cual se anula el programa de Becas Préstamo y se establece la modalidad de beca propiamente dicha, el cual sigue vigente en la actualidad según el Manual de Organización.

Actualmente la Sección Socioeconómica tiene un archivo físico de gran tamaño, el cual contiene todos los expedientes físicos de las solicitudes de becas que han solicitado los estudiantes. Hace unos años se había iniciado la digitalización de estos archivos, con el objetivo de facilitar el proceso de búsqueda y gestión de las solicitudes, pero debido a un daño en el ordenador que contenía los expedientes, se perdió la información.

Debido a la situación actual del proceso de las solicitudes de beca, se ha solicitado al Departamento de Procesamiento de Datos de la Universidad de San Carlos de Guatemala un módulo que facilite la recepción de documentos requeridos por parte de los estudiantes, y las demás gestiones de informes durante el proceso de la solicitud.

1.1.2. Misión

Ser el responsable de realizar el estudio de la situación socioeconómica del estudiante universitario, de escasos recursos económicos y alto rendimiento académico, que solicita una beca de pregrado, en la Universidad de San Carlos de Guatemala. Así mismo, atender solicitudes de exoneraciones parciales en el pago de matrícula a estudiantes extranjeros. (USAC, 1959, s.f.)

1.1.3. Visión

“Ser el ente capaz de responder a la demanda de toda la población estudiantil universitaria, de escasos recursos económicos y alto rendimiento académico, que aspiran a recibir la beca universitaria que le permita concluir satisfactoriamente sus estudios superiores” (USAC, 1959, s.f.).

1.1.4. Servicios que realiza.

Según la Sección Socioeconómica, Universidad de San Carlos de Guatemala (1959):

a. Administrar el Programa de Becas de Pregrado de la Universidad de San Carlos de Guatemala.

b. Proporcionar becas por subvención económica.

c. Nivelar la cobertura de becas equitativamente a todas las unidades académicas.

d. Gestionar el incremento del presupuesto para el programa de becas.

e. Agotar el presupuesto de becas asignado anualmente.

f. Realizar acciones dirigidas al sector público y privado, a fin de que permita incrementar el fondo de ayudas económicas, que la Universidad de San Carlos de Guatemala asigna presupuestalmente para becas.

g. Brindar becas de alojamiento al estudiante universitario.

h. Administrar y supervisar el alojamiento en la “Casa del Estudiante”.

i. Promover proyectos académicos, deportivos, recreativos y culturales, para propiciar la formación integral del estudiante becado.

j. Estimular al estudiante becado en la realización del servicio social, como labor de extensión de la universidad.

k. Apoyar y promover la creación y desarrollo de programas de apoyo socioeconómico y apoyo académico a estudiantes de pregrado, por parte de las unidades académicas de la universidad y de las asociaciones estudiantiles.

l. Coordinar con la Unidad de Salud asistencia especial al grupo de estudiantes becados.

m. Coordinar con las distintas dependencias de la universidad, acciones que involucren al estudiante becado en programas de servicio social.

n. Emitir dictámenes técnicos que sirvan de base para la toma de decisiones de qué estudiantes llenan los requisitos para ser alojados en la residencia universitaria.

o. Elaborar y evaluar el Plan Operativo Anual de la Sección Socioeconómica. (s.f.)

1.2. Descripción de las necesidades

Las necesidades se identificaron por medio de reuniones con la Licenciada Patricia García jefa de la Sección Socioeconómica, Licenciada Elizabeth Ramírez secretaria de comisión y el Licenciado Rodrigo Mendizábal Analista de Cómputo y Coordinador de Proyectos Informáticos del Departamento de Procesamiento de Datos, estas se describen a continuación:

- Desarrollar una aplicación web que permita la gestión de la documentación requerida a los estudiantes, el manejo de informe domiciliar, informe de diagnóstico y expediente completo, y la gestión de archivos.

- Solicitar un servidor al Departamento de Procesamiento de Datos de la Universidad de San Carlos de Guatemala para poner en producción la aplicación web y que se almacenen todas las solicitudes y sus documentos.

- Identificar las herramientas de desarrollo para la aplicación web, teniendo en cuenta que el sistema operativo del servidor es Ubuntu LTS 16.04, y que sean de código abierto con el objetivo de evitar el pago de licencias de tal manera que no afecte el presupuesto de la Sección Socioeconómica.

- Crear documentación de usuario como documentación técnica.

- Proporcionar capacitación al personal de la Sección Socioeconómica para el uso correcto de la aplicación web.

1.3. Priorización de las necesidades

Para dar prioridad a las necesidades planteadas se ha utilizado la técnica MoSCoW la cual se caracteriza por ser esencial en el método de desarrollo de sistemas dinámicos. Esta técnica le asigna una de las letras a las características a considerar en la priorización, estas se describen a continuación:

- M (Must have) Debe tener: requisito obligatorio que debe estar implementado, ya que sin este el producto final no sería funcional.

- S (Should have) Debería tener: requisito de alta prioridad que debe estar implementado, aunque se podría prescindir de este si fuera necesario en el producto final.

- C (Could have) Podría tener: requisito que no es necesario y su ausencia no será perjudicial en el producto final.

- W (Won't have) No tendrá: requisito que están descartados o que son de baja prioridad, que en un futuro se podrían implementar, y su ausencia no afectará el producto final.

Tabla I. **Identificación de necesidades**

Necesidad	Categoría
Modificación al formulario de solicitud de beca en la plataforma SIIF.	W
Diseño, creación e implementación de la base de datos en MySQL.	M
Implementación de servidor de archivos.	W
Sistema de autenticación para todos los usuarios involucrados.	M
Gestión de la documentación requerida y posterior tanto para los estudiantes como para los trabajadores sociales.	M

Continuación de la tabla I.

Gestión del informe domiciliar, informe diagnóstico y expediente completo por solicitud.	M
Gestión de cuadros de propuesta.	M
Gestión de acuerdos de adjudicación y convenios.	M
Módulo de reportes para la secretaria de comisión y trabajador social.	M
Creación, modificación y eliminación de documentos posteriores.	S
Creación de documentación de usuario y técnico.	M
Activación de la petición de solicitudes de becas al servidor de Procesamiento de Datos.	C

Fuente: elaboración propia.

1.4. Análisis FODA

Conocer las fortalezas, debilidades, oportunidades y amenazas es de suma importancia al inicio del planteamiento de este proyecto, para determinar los alcances de este, los cuales se describen a continuación.

1.4.1. Fortalezas

- Se cuenta con el acceso a Control de Correspondencia que se utilizará como base para implementar la misma esencia hacia el nuevo módulo.

- La sección socioeconómica cuenta con el apoyo del Departamento de Procesamiento de Datos de la Universidad de San Carlos de Guatemala para la implementación del nuevo módulo.

- La documentación que se requiere para cada beca está bien definida.

- El personal que labora en la sección socioeconómica es cooperativo y tienen bien definidas sus funciones en los requerimientos que se solicitan.

- Autonomía en el uso de lenguajes y herramientas de programación para el desarrollo del módulo.

1.4.2. Debilidades

- Posee un presupuesto ajustado de acuerdo con las necesidades que suple la Sección Socioeconómica de la Universidad de San Carlos de Guatemala.

- La plataforma que se utiliza actualmente en la sección socioeconómica requiere de varios cambios.

- Demora en los cambios que se requieran realizar en el módulo por falta de mantenimiento.

- El Departamento de Procesamiento de Datos no permita cierto grado de independencia con la Sección Socioeconómica.

1.4.3. Oportunidades

- El nuevo módulo permitirá agilizar las funcionalidades que anteriormente en la sección socioeconómica se realizaban de forma manual.

- El archivo digital de cada solicitud de beca se registrará y almacenará en un servidor propio de la sección socioeconómica.

- El nuevo módulo permitirá expandir sus funcionalidades y en un futuro permitir una migración del sistema anterior que utilizaba la sección socioeconómica.

- El nuevo módulo permitirá que las solicitudes incompletas se descarten de forma automática.

1.4.4. Amenazas

- Demora en la obtención de los servicios que se requieran consumir desde el Departamento de Procesamiento de Datos.

- El archivo físico de solicitudes de becas es de gran tamaño que dificulta el proceso de búsqueda de algún expediente en específico.

- El personal de trabajo de la sección no realice las verificaciones correctas de los documentos requeridos y el proceso de la solicitud de beca.

- Al depender del Departamento de Procesamiento de Datos las solicitudes de becas se demoran meses en completarse.

2. FASE TÉCNICO PROFESIONAL

2.1. Software

Es un conjunto de instrucciones, componentes lógicos, procedimientos o rutinas que son independientes del hardware y permite que un sistema informático sea programable con el objetivo de realizar tareas específicas. Existen tres tipos de software, estos son:

- Software del sistema: conjunto de programas que se utilizan como intermediario entre el hardware y el usuario.

- Software de programación: conjunto de programas que proporcionan herramientas como editores de texto, compiladores, depuradores, entre otros.

- Software de aplicación: es un programa o conjunto de programas que realizan una tarea específica, que puede estar relacionada con la productividad, creatividad o comunicación.

2.2. Aplicación de escritorio

Es un software o programa que se puede instalar en un sistema informático y que un usuario puede ejecutar de manera local para realizar una tarea específica. Este programa ocupa espacio en el disco duro del sistema informático y en general estas pueden ejecutarse sin tener conexión a internet.

2.2.1. Características

- Requiere instalación en la computadora para ejecutarse.
- Únicamente puede ser accedido por los usuarios que están en la computadora.
- No requiere una conexión a internet para ejecutarse.
- Requiere de espacio en el disco duro en la computadora.
- Las aplicaciones están desarrolladas para ejecutarse en un sistema operativo específico.
- Las actualizaciones deben realizarse individualmente en cada equipo donde se haya instalado.
- Para ejecutarse correctamente se requiere que cumplan ciertos requisitos de hardware como la arquitectura, espacio disponible en el disco duro, memoria RAM, entre otros.

2.3. Aplicación web

Es un software o programa que se puede ejecutar desde cualquier navegador web por medio de una URL específica. Esta aplicación cuenta con el lado del cliente o frontend el cual ejecuta la aplicación y cuenta con el lado del servidor o backend el cual se encarga de ejecutar las peticiones del cliente.

2.3.1. Características

- Requiere instalación únicamente en el servidor web.
- Cualquier usuario puede acceder a la aplicación y no requiere instalación.
- Puede ser accedida desde cualquier lugar a través de internet.
- Únicamente ocupa espacio en el servidor web.
- Puede ejecutarse desde cualquier sistema operativo.
- Las actualizaciones únicamente se realizan en el servidor web.
- Es independiente del hardware ya que para ejecutarse únicamente requiere un navegador web y una conexión a internet.

2.4. Diferencia entre sitio web y aplicación web

Toda aplicación web es un sitio web, ante esta afirmación surge la interrogante acerca de la diferencia de estos términos. A continuación, se muestra una tabla con sus principales diferencias.

Tabla II. **Diferencias entre sitio web y aplicación web**

Sitio web	**Aplicación web**
El sitio web tiene contenido informativo y es estático.	La aplicación web tiene contenido informativo y es dinámico.

Continuación de la tabla II.

Sitio web	**Aplicación web**
El usuario puede leer el contenido del sitio web, pero no tiene ninguna interacción.	El usuario puede leer el contenido del sitio web y también puede interactuar con este.
La información es de acceso público	La información está restringida y solo puede ser accedida por usuarios registrados.
No requiere de autenticación para su uso.	Requiere de autenticación para su uso.
El sitio web no necesita ser precompilado.	La aplicación web debe estar precompilada antes de su implementación.
Los cambios no requieren una recompilación completa, solo se necesita actualizar el código HTML.	Los cambios requieren que el proyecto se tenga que compilar e implementar de nuevo.

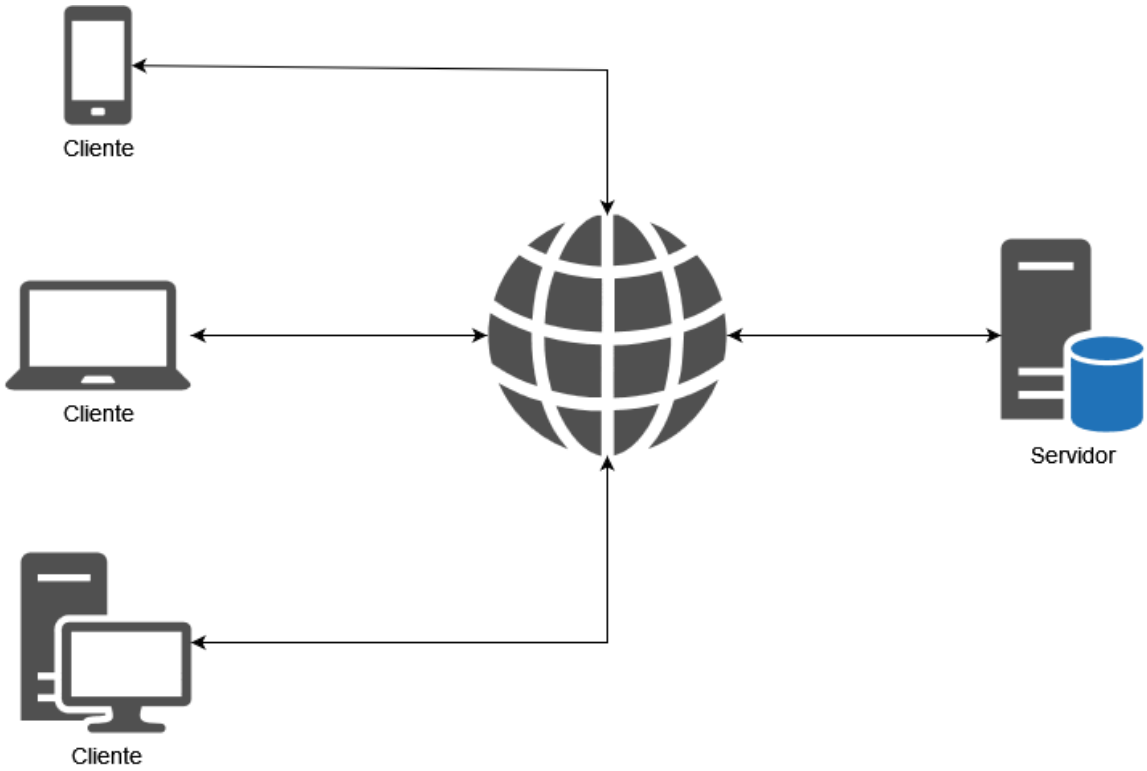
Fuente: elaboración propia.

2.5. Selección de herramientas para el desarrollo

Teniendo como base las características que implican el desarrollo de una aplicación de escritorio, una de las principales desventajas es el uso limitado de usuarios ya que en teoría únicamente un usuario puede utilizar la aplicación en la computadora donde se realizó la instalación, esto además implica que al momento de realizar alguna actualización se debe realizar en todas las computadoras que tienen la aplicación.

Al conocer la diferencia entre un sitio web y una aplicación web, se ha identificado que la aplicación web es esencial para uso de tareas complejas, que no limita el acceso de usuarios, además de no depender del sistema operativo y tener en cuenta que al momento de realizar una modificación únicamente se aplicará en el servidor web. A continuación, se muestra un diagrama con la arquitectura del sistema.

Figura 1. **Arquitectura cliente servidor**



Fuente: elaboración propia, realizado con draw.io.

2.6. Frontend

Se refiere a los elementos visuales como texto, botones, imágenes, entre otros. o interfaz de usuario, es la parte de la aplicación que permite al usuario ver los elementos visuales e interactuar con las funcionalidades. Es el conjunto de tecnologías de diseño y desarrollo que se ejecutan en el navegador, también es común referirse al frontend como el lado del cliente de una aplicación.

El desarrollo del frontend se construye utilizando las siguientes tecnologías:

- HTML: lenguaje de marcado de hipertexto (HyperText Markup Language) es el código estándar que se utiliza para crear páginas web.

- CSS: hojas de estilo en cascada (Cascading Style Sheets) es el código que permite aplicar estilos a las páginas web.

- Javascript: lenguaje de programación interpretado que se utiliza para que la aplicación web sea interactiva con el usuario.

2.7. Backend

Se refiere a la parte que está en el lado del servidor de la aplicación, que recibe las solicitudes del cliente, la cual no entra en contacto directo con el usuario. Es el conjunto de instrucciones que se ejecutan en el servidor y se encarga del acceso, almacenamiento y organización de los datos.

También es común referirse al backend como el lado del servidor de una aplicación. La implementación del backend está conformado por tres partes:

- Servidor: es la computadora que se encarga de recibir las solicitudes del cliente.

- Aplicación: es el conjunto de instrucciones que se ejecuta en el servidor y es el responsable de escuchar y responder las solicitudes.

- Base de datos: es el programa que tiene la responsabilidad de almacenar los datos de forma persistente en el almacenamiento del servidor.

2.8. Web Service

Es un conjunto de protocolos y estándares que permiten la comunicación e intercambio de datos entre diferentes sistemas o aplicaciones.

2.8.1. Características de un web service

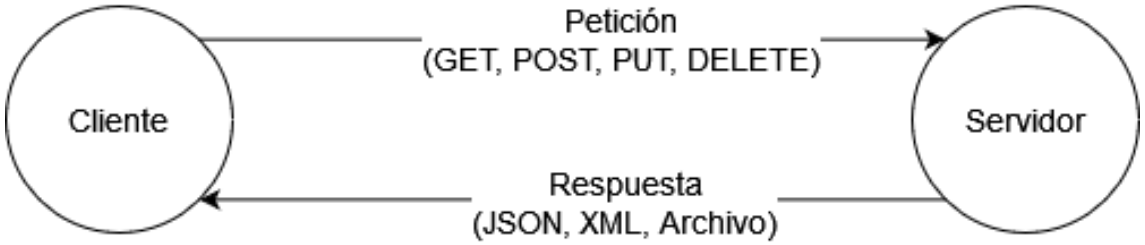
- Está disponible a través de internet o por redes privadas.
- Es independiente de cualquier tipo de sistema operativo o lenguaje de programación.
- Es detectable por medio de un mecanismo de búsqueda simple.
- Utiliza un protocolo de mensajería XML.

2.9. REST

Transferencia de Estado Representacional (*Representational state transfer)* es un estilo de arquitectura de software que define un conjunto de restricciones, que se utilizan para crear *web services*. Esta tecnología se caracteriza por ser ligera, simple y flexible.

La siguiente figura muestra la interacción que se produce entre el lado del cliente y el lado del servidor por medio la API REST.

Figura 2. **Representación API REST**



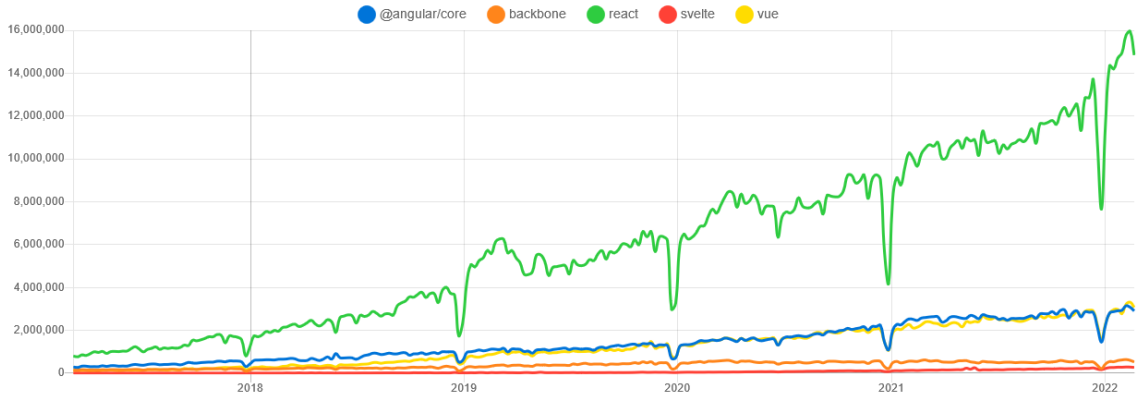
Fuente: elaboración propia, realizado con draw.io.

2.10. Herramientas para el desarrollo de Frontend y Backend

Para el desarrollo del frontend es necesario implementar HTML, CSS y Javascript, debido a que la creación de una aplicación web es más compleja en comparación a la de un sitio web, existen herramientas que facilitan, simplifican y reducen el riesgo de errores durante el desarrollo del proyecto, comúnmente conocidos como frameworks, los más populares en la actualidad son: Angular, React, Vue.js, Backbone.js y Svelte.

La siguiente imagen muestra la cantidad de descargas de frameworks para fronted que se ha realizado en los últimos 5 años.

Figura 3. **Descargas de frameworks frontend populares en los últimos 5 años**

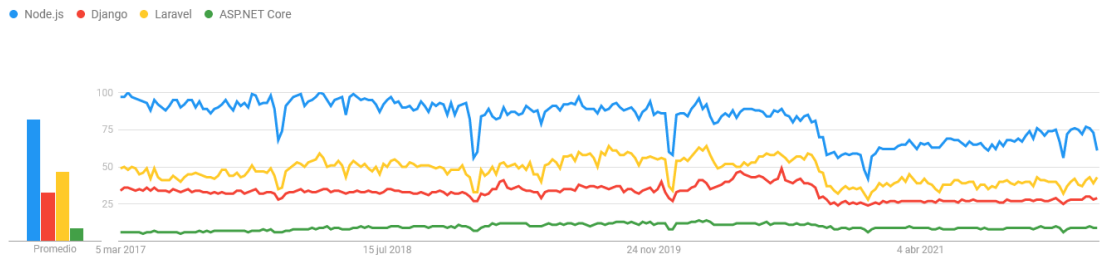


Fuente: NPM trends (2022). *@angular/core vs backbone vs react vs svelte vs vue.* Consultado febrero de 2022. Recuperado de https://www.npmtrends.com/@angular/core-vs-backbone-vs-react-vs-svelte-vs-vue.

Para el desarrollo del backend se cuentan con bastantes lenguajes de programación que permiten el desarrollo y creación de la aplicación encargada de escuchar y responder las solicitudes. Al igual que con el frontend, existen frameworks que facilitan el desarrollo y cubren áreas como integración, seguridad y escalabilidad, los más populares son: Node.js, Django, Laravel y ASP.NET Core.

La siguiente imagen muestra la cantidad de descargas de frameworks para backend que se ha realizado en los últimos 5 años.

Figura 4. **Descargas de frameworks backend populares en los últimos 5 años.**



Fuente: Google Trends (2022). *Node.js, Django, Laravel, ASP.NET Core*. Consultado febrero de 2022. Recuperado de https://trends.google.com/trends/explore?date=today%205-y&q=%2Fm%2F0bbxf89,%2Fm%2F06y_qx,%2Fm%2F0jwy148,%2Fm%2F0115qhwr.

2.11. Angular

Es un framework y una plataforma de desarrollo, escrito en TypeScript, que se utiliza para el desarrollo de aplicaciones web de una sola página (Single Page Application) o aplicaciones web progresivas (Progressive Web App), este cuenta con soporte por un equipo de Google y una gran comunidad de desarrolladores. (Angular, 2022)

Este framework permite que se ahorre tiempo de desarrollo debido a que utiliza el patrón modelo vista controlador (MVC). Angular ofrece aplicaciones más rápidas y ligeras, este framework se basa en componentes con el objetivo de crear aplicaciones escalables y mantenibles.

2.11.1. Ventajas de Angular Material

- Menor tiempo de desarrollo.

- Documentación.
- Seguridad.
- Multiplataforma.
- Escalabilidad
- Fácil implementación de pruebas.
- Comunidad.

2.12. Angular Material

Es una biblioteca de interfaz de usuario, desarrollado por el equipo de Google, que se utiliza en proyectos de Angular para desarrollar interfaces de usuario más elegantes y consistentes. (Angular Material, s.f.) Esta biblioteca se utiliza para crear aplicaciones web funcionales, atractivas, rápidas y responsivas, por medio de componentes como botones, campos de texto, tablas, entre otros.

2.12.1. Características de Angular Material

- Instalación fácil.
- Brinda una gran cantidad de componentes útiles, que son fáciles de implementar.
- Versátil.

- Cuenta documentación detallada y una comunidad.

- Compatible con múltiples dispositivos, ya que tiene un diseño responsive.

2.13. Node.js

Es un entorno de ejecución de código abierto que utiliza el motor Javascript V8 de Google Chrome para ejecutar código Javascript fuera del navegador. (Node.js, s.f.) Con el tiempo ha adquirido bastante aceptación debido a la popularidad del lenguaje de programación Javascript y la amplia utilidad que brinda al poder desarrollar todo tipo de aplicaciones, tales como aplicaciones de línea de comandos, aplicaciones de chat, transmisión de datos, aplicaciones de una sola página y mayormente utilizada para crear servidores web.

2.13.1. Ventajas de Node.js

- Es publicado bajo la licencia MIT.

- Es utilizado por grandes corporaciones, lo que acredita la seguridad y confianza para utilizar esta herramienta.

- Alto rendimiento y ejecución rápida del código, y esto se debe a que utiliza el motor Javascript V8 de Google.

- Escalabilidad, tanto vertical como horizontal.

- Cuenta con una destacada biblioteca de paquetes, npm, la cual contiene más de un millón de bibliotecas que agilizan el desarrollo de aplicaciones.

- Compatibilidad multiplataforma.
- Amplia documentación y soporte por parte de una comunidad.
- Es fácil de aprender.
- Es compatible con la arquitectura MVC (Modelo Vista Controlador).
- Se puede desarrollar tanto el Frontend como el Backend de una aplicación web.

2.14. NPM

NPM (Node Package Manager) es el registro de software más grande del mundo, que tiene como función ser el administrador de paquetes para Node. (NPM, s.f.) Este consta de tres componentes:

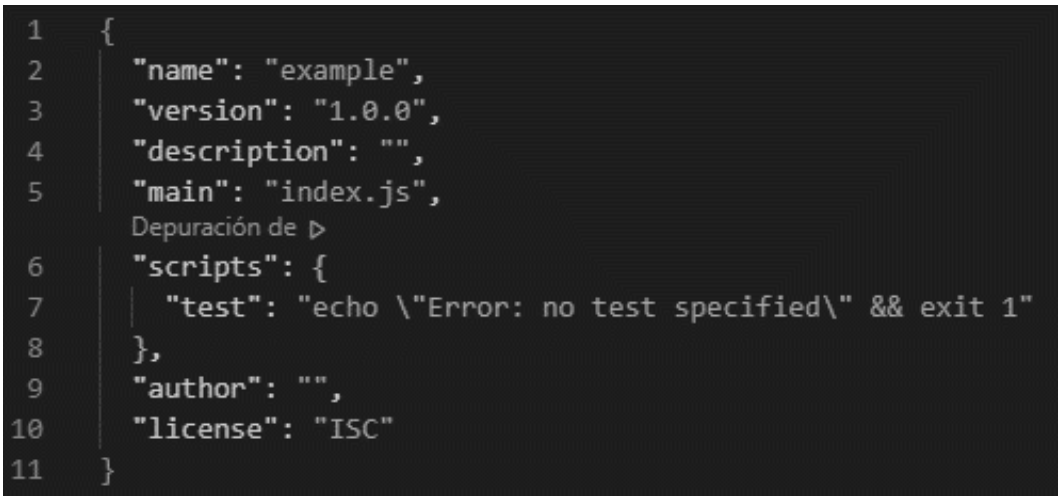
- Sitio web, dónde se puede consultar y encontrar paquetes, tanto de acceso público como privado.
- CLI (Command Line Client) es el cliente de línea de comandos que se utiliza para interactuar con npm, por ejemplo, la descarga de paquetes.
- El registro, que es una base de datos pública que contiene todos los paquetes de software libre publicados y a los que se puede acceder.

2.14.1. Implementación

Antes de crear un proyecto se debe instalar Node.js, después se dirige a la carpeta donde se desea gestionar el entorno de desarrollo. Desde la terminal se escribe el comando ‘npm init’ e inmediatamente se le mostrará la configuración del archivo package.json, este archivo se encarga de indicarle al administrador que el directorio donde se ubica el archivo es un proyecto o un paquete npm. El archivo package.json contiene el nombre del proyecto, la versión inicial, descripción, punto de entrada, palabras clave, definición de comandos, repositorio de git, licencia, dependencias y dependencias de desarrollo.

La descarga e instalación de paquetes se realizarán con el comando ‘npm install <nombre_paquete>’, y el cambio se verá reflejado en el archivo package.json en el apartado de dependencias.

Figura 5. **Estructura básica de un archivo package.json**



Fuente: npm Docs (2022). *package.json.* Consultado junio de 2022. Recuperado de https://docs.npmjs.com/cli/v8/configuring-npm/package-json.

2.15. Express.js

Es un framework de aplicación web de Node.js, este se caracteriza por ser mínimo y flexible, el cual brinda un conjunto de funciones que se utiliza para crear aplicaciones web del lado del servidor de manera rápida y sencilla. (Express, 2017)

2.15.1. Características de Express.js

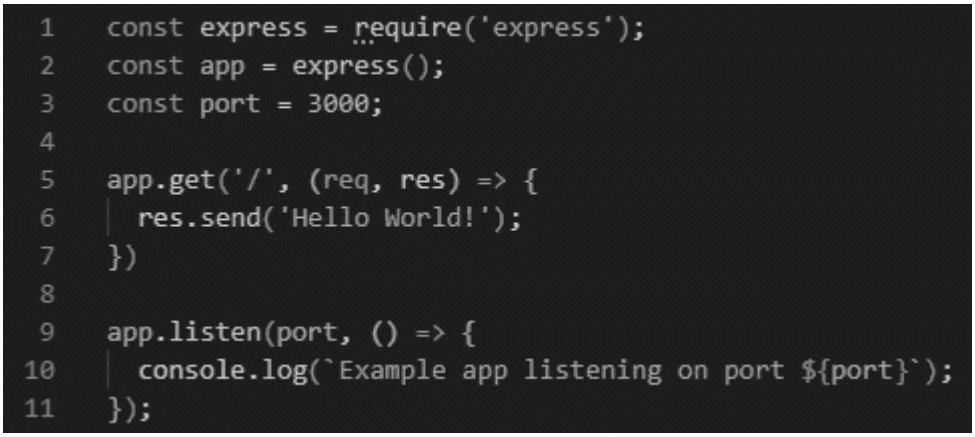
- Desarrollo de un servidor web rápido.
- Facilidad de configuración y personalización.
- Facilita la organización de la funcionalidad con el middleware y enrutamiento.
- Permite definir y responder rutas de aplicación que se basan en métodos HTTP y URL específicas.
- La conexión a bases de datos es simple y fácil de implementar.
- Uso de middleware, permite implementar un controlador que tiene acceso al ciclo de la solicitud y respuesta de la aplicación.
- Es compatible con todo tipo de Sistemas Operativos que sean compatibles con Node.js.
- Es compatible con la arquitectura MVC (Modelo Vista Controlador).

- Se actualiza constantemente debido a su amplia comunidad de código abierto.

2.15.2. Implementación

Primero se debe instalar el paquete con el comando ‘npm install express’ estando en el directorio que contiene el archivo package.json. Para crear un servidor web debe dirigir al archivo principal el cual se indicó como punto de entrada, agregando al inicio de este la importación de Express y posteriormente crear una variable la cual servirá para agregar todas las configuraciones que sean necesarias, tales como número de puerto, configuración del middleware, métodos de enrutamiento, endpoint o ruta, método de solicitud HTTP (GET, POST, DELETE, entre otros), vinculación al puerto de escucha del servidor, entre otras. Luego de finalizar la configuración, se debe correr el servidor por medio de la terminal utilizando el siguiente comando “node <nombre_archivo>.js”.

Figura 6. **Implementación de un servidor web**



```
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => {
  res.send('Hello World!');
})

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`);
});
```

Fuente: Express (2022). *Ejemplo “Hello world”.* Consultado junio de 2022. Recuperado de https://expressjs.com/es/starter/hello-world.html.

2.16. Base de datos

Es una colección sistemática u organizada de información estructurada, o datos, que se almacena en un sistema informático. Estos datos se organizan en forma de tablas, vistas, informes, entre otros. Una base de datos está controlada por un sistema de administración de base de datos, el cual permite acceder, recuperar, administrar y actualizar datos.

2.17. Proceso de solicitud de beca

El proceso actual de solicitud de beca se describe a continuación:

- La Sección Socioeconómica habilita la recepción de solicitudes desde la plataforma SIIF. Los estudiantes interesados realizan el trámite desde la plataforma y esperan hasta que se les asigne una oficina.

- La secretaria de recepción le notifica al estudiante por medio de correo electrónico, el número de oficina, los días hábiles para entregar la documentación requerida y la fecha para la entrevista.

- El estudiante envía por correo electrónico la documentación requerida y se presenta en la oficina de la Sección Socioeconómica para su entrevista.

- El trabajador social realiza la entrevista al estudiante, si su documentación requerida está en orden, se procede a realizar el siguiente paso. En caso de que algún documento no sea válido, el trabajador social debe solicitar correcciones al estudiante.

- El trabajador social solicita otros documentos.

- El estudiante le envía por correo electrónico al trabajador social los documentos posteriores.

- El trabajador social realiza la revisión de la documentación posterior, sí todo está en orden se procede a realizar el siguiente paso. En caso de que algún documento no sea válido, el trabajador social debe solicitar correcciones al estudiante.

- El trabajador social realiza la visita domiciliar y elabora informe.

- El trabajador social elabora diagnóstico y lo traslada a secretaria de comisión, junto con el expediente completo.

- Secretaria de comisión revisa el expediente. El expediente completo comprende en los dos informes y todos los documentos solicitados al estudiante.

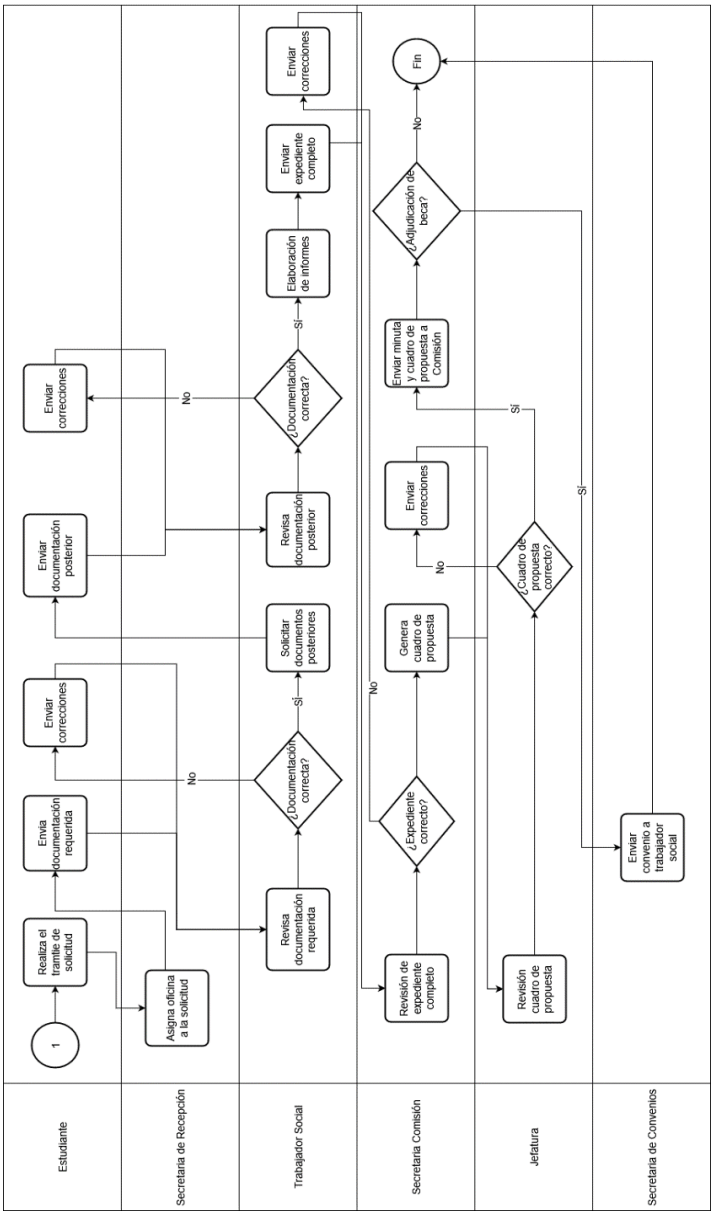
- Secretaria de comisión calcula promedio e ingresa al sistema, posteriormente genera el cuadro de propuesta y traslada a jefatura para revisión.

- Jefatura revisa y solicita correcciones.

- Secretaria de comisión envía minuta y cuadro de propuesta a Comisión.

En este proceso se involucran varios actores, estos son: estudiante, secretaria de recepción, trabajador social, secretaria de comisión, jefatura y secretaria de convenios. El siguiente diagrama representa con más detalle la secuencia de la solicitud de beca y todos los actores implicados.

Figura 7. **Secuencia solicitud de beca**

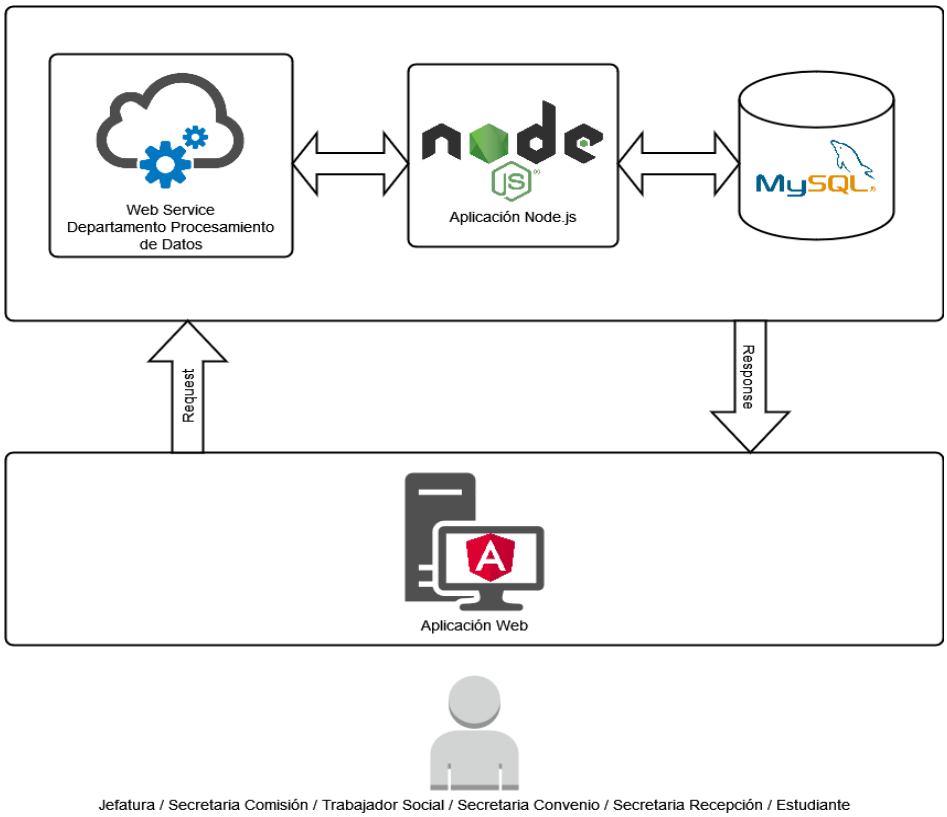


Fuente: elaboración propia, realizado con draw.io.

2.18. Preliminares de la solución

Al describir todas las tecnologías que conlleva la creación de una aplicación web, y presentar las ventajas de ciertas herramientas para cada elemento, se seleccionó el framework Angular para el desarrollo del frontend, el entorno de ejecución Node.js para el desarrollo del backend y MySQL como sistema de gestión de base de datos. En la siguiente imagen se muestra la arquitectura que se implementó para el desarrollo de este proyecto. Los diagramas de casos de uso para cada rol en el sistema se presentan en la sección de apéndice en este documento.

Figura 8. **Arquitectura de la solución**



Fuente: elaboración propia, realizado con draw.io.

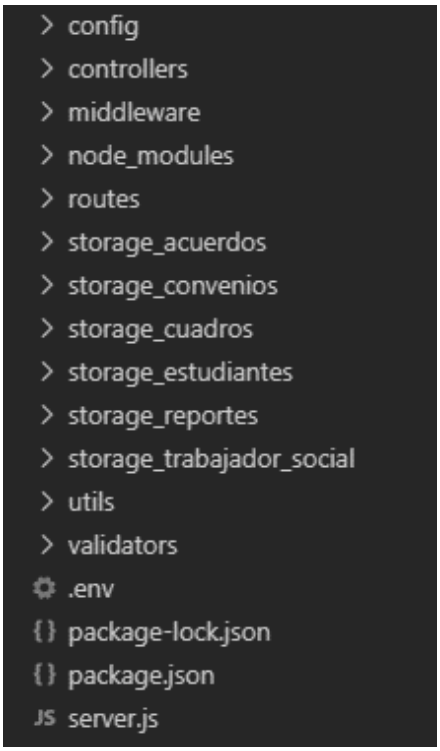
2.19. Solución del proyecto

El desarrollo del backend y frontend se describen en las siguientes secciones de este documento.

2.19.1. Servidor del proyecto

El servidor se implementó por medio del manejador de paquetes de Node.js (npm), el patrón de diseño Modelo Vista Controlador (MVC) se utilizó para la estructura de las carpetas y archivos del servidor, con el único detalle que la interfaz de usuario o vista se reemplazó por las rutas de acceso a las peticiones del servidor. En la siguiente imagen se muestra la estructura completa.

Figura 9. **Estructura de carpetas del servidor**



Fuente: elaboración propia, realizado con Visual Studio Code.

2.19.1.1. package.json

Archivo encargado de indicarle al sistema que es un proyecto de Node.js, el cual contiene la información del servidor como el nombre, versión, descripción, punto de entrada, definición de comandos, autor, licencia y dependencias.

2.19.1.2. package-lock.json

Archivo encargado de describir el árbol de dependencias del proyecto, este se genera automáticamente al momento de instalar las dependencias del proyecto o cuando ocurre algún cambio en el archivo package.json, de tal manera que se pueda instalar las mismas versiones de todos los paquetes donde se desarrolló el proyecto.

2.19.1.3. server.js

Archivo que se indicó como punto de entrada en el archivo package.json, este archivo es el encargado de crear y ejecutar el servidor web, además este archivo contiene las funcionalidades para ejecutarse a determinada hora una petición al servicio proporcionado por el Departamento de Procesamiento de Datos y permitir que pueda realizar la automatización de la obtención de solicitudes y la denegación de solicitudes.

2.19.1.4. .env

Archivo que contiene todas las variables de entorno del backend, esto se utiliza como buena práctica de programación para manejar la seguridad de ciertos valores que no pueden estar escritos directamente en los archivos de desarrollo.

2.19.1.5. node_modules

Directorio que contiene todos los paquetes de dependencia que el servidor utiliza, los más relevantes que se utilizaron se describen a continuación.

2.19.1.5.1. Cors

Paquete de Node.js que se utiliza para establecer las políticas el intercambio de recursos de origen cruzado. Su implementación es sencilla, se importa y se le indica a Express que se va a utilizar.

Figura 10. **Implementación cors**

```
1   const express = require('express');
2   const cors = require('cors');
3
4   const app = express();
5   app.use(cors());
```

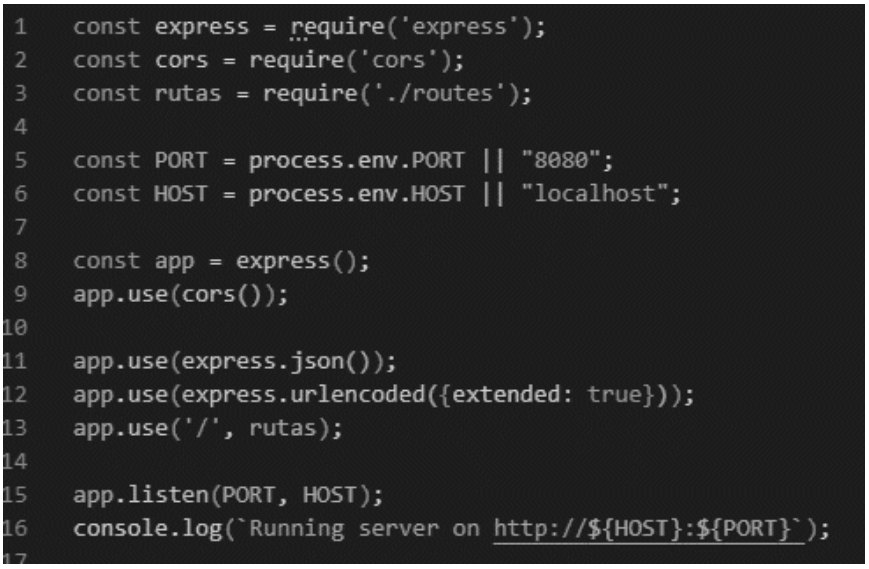
Fuente: elaboración propia, realizado con Visual Studio Code.

2.19.1.5.2. Express

Paquete de Node.js que se utiliza para crear el servidor web. Su implementación requiere de su importación y posteriormente indicarle las configuraciones que se necesiten. En la siguiente imagen se puede observar que se le indica a Express el uso de cors, también se le indica que debe utilizar una función middleware que analiza las solicitudes entrantes con cargas JSON y urlencoded la cual se basa en body-parser. Se hace la extracción de las rutas de todos los archivos que están contenidos en el directorio routes y se le asignan a

Express. Por último, se le indica que el servidor debe estar escuchar las peticiones en cierto puerto y host.

Figura 11. **Implementación express**



```
const express = require('express');
const cors = require('cors');
const rutas = require('./routes');

const PORT = process.env.PORT || "8080";
const HOST = process.env.HOST || "localhost";

const app = express();
app.use(cors());

app.use(express.json());
app.use(express.urlencoded({extended: true}));
app.use('/', rutas);

app.listen(PORT, HOST);
console.log(`Running server on http://${HOST}:${PORT}`);
```

Fuente: elaboración propia, realizado con Visual Studio Code.

2.19.1.5.3. Express Validator

Paquete de Node.js que se utiliza para validar que los parámetros entrantes de la petición sean los solicitados, que el tipo de dato de cada parámetro sea correcto, entre otras validaciones, en caso contrario se podrá rechazar la petición.

Figura 12. **Implementación reglas express validator**

```
const { check } = require('express-validator');
const { validarResultados } = require('../utils/handleValidator');

const validadorCarga = [
    check("data").exists().notEmpty().isObject(),
    (req, res, next) => {
        validarResultados(req, res, next);
    },
];
```

Fuente: elaboración propia, realizado con Visual Studio Code.

Figura 13. **Implementación validación express validator**

```
const { validationResult } = require('express-validator');

const validarResultados = (req, res, next) => {
    try{
        // Valida el request y si existe un error lanza una excepción.
        validationResult(req).throw();
        // Si no existe error continua hacia el controlador.
        return next();
    }
    catch(err){
        res.status(400).send({message: "Se han ingresado campos inválidos."});
    }
}
```

Fuente: elaboración propia, realizado con Visual Studio Code.

2.19.1.5.4. JSON Web Token

Paquete de Node.js que se utiliza para crear tokens encriptados, este consta de tres partes codificadas, la primera parte es el encabezado que contiene el tipo de algoritmo de encriptación, la segunde parte contiene los datos o información que se desea incorporar, y la última parte es la firma que permite verificar que el token es válido. (Okta, s.f.)

La autenticación para este proyecto se implementó con este paquete en el lado del servidor, de tal manera que cualquier usuario que inicie sesión con credenciales validas le sea generado un token de sesión y un refreshtoken. La descripción del proceso de autenticación se describe a continuación:

- El usuario ingresa a la aplicación y envía una petición con sus credenciales.

- El servidor verifica que las credenciales son válidas, si este fuera el caso entonces el servidor genera el token de sesión y el refreshtoken y los envía como respuesta, como buena práctica de seguridad se recomienda que el token de sesión tenga una vigencia corta y el refreshtoken tenga una vigencia mayor. Si las credenciales no son correctas se responde al cliente que el usuario o contraseña son incorrectos.

- El cliente recibe los tokens y los almacena, es recomendable utilizar cookies y evitar el localstorage como medio de almacenamiento.

- El cliente realiza una petición y adjunta en el header el token de sesión y el refreshtoken.

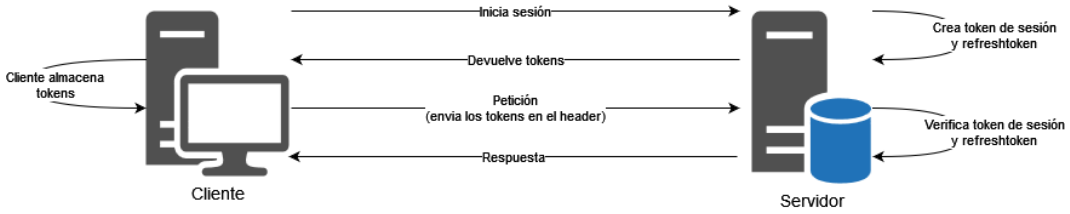
- El servidor verifica que el token de sesión es válido, si no es válido rechaza la petición y responde con código de estado de respuesta HTTP 401. Si el token es válido responde la solicitud del cliente.

- El cliente obtiene la respuesta, si la petición fue rechazada con código 401 el cliente realizará una petición al servidor para verificar el refreshtoken.

- El servidor recibe la petición y verifica si el refreshtoken es válido, si es inválido se rechaza la petición y se responde con código de estado de respuesta HTTP 400. Si el refreshtoken es válido se verifica que su tiempo de expiración sea menor de un día, si es así se actualiza el refreshtoken para que el usuario no vuelva a iniciar sesión, en caso contrario se retorna el nuevo token de sesión y el refreshtoken.

- El cliente obtiene la respuesta, si la petición fue rechazada con código 400 se cierra la sesión automáticamente y el usuario deberá iniciar una sesión nueva, en caso contrario se almacenan los nuevos tokens y se repite automáticamente la petición con los tokens válidos.

Figura 14. **Implementación JWT**

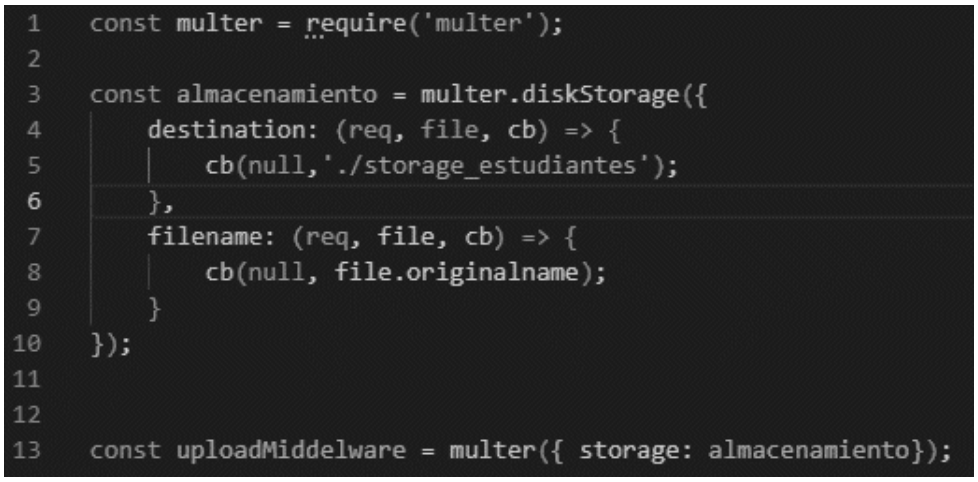


Fuente: elaboración propia, realizado con draw.io.

2.19.1.5.5. Multer

Paquete de Node.js que se utiliza para manejar datos del multipart/form-data en Express cuando los usuarios tienen que adjuntar archivos en las peticiones. Su implementación es simple, se requiere su importación y posteriormente se hace la configuración del almacenamiento y nombre de los archivos que se almacenarán en el servidor.

Figura 15. **Implementación multer**



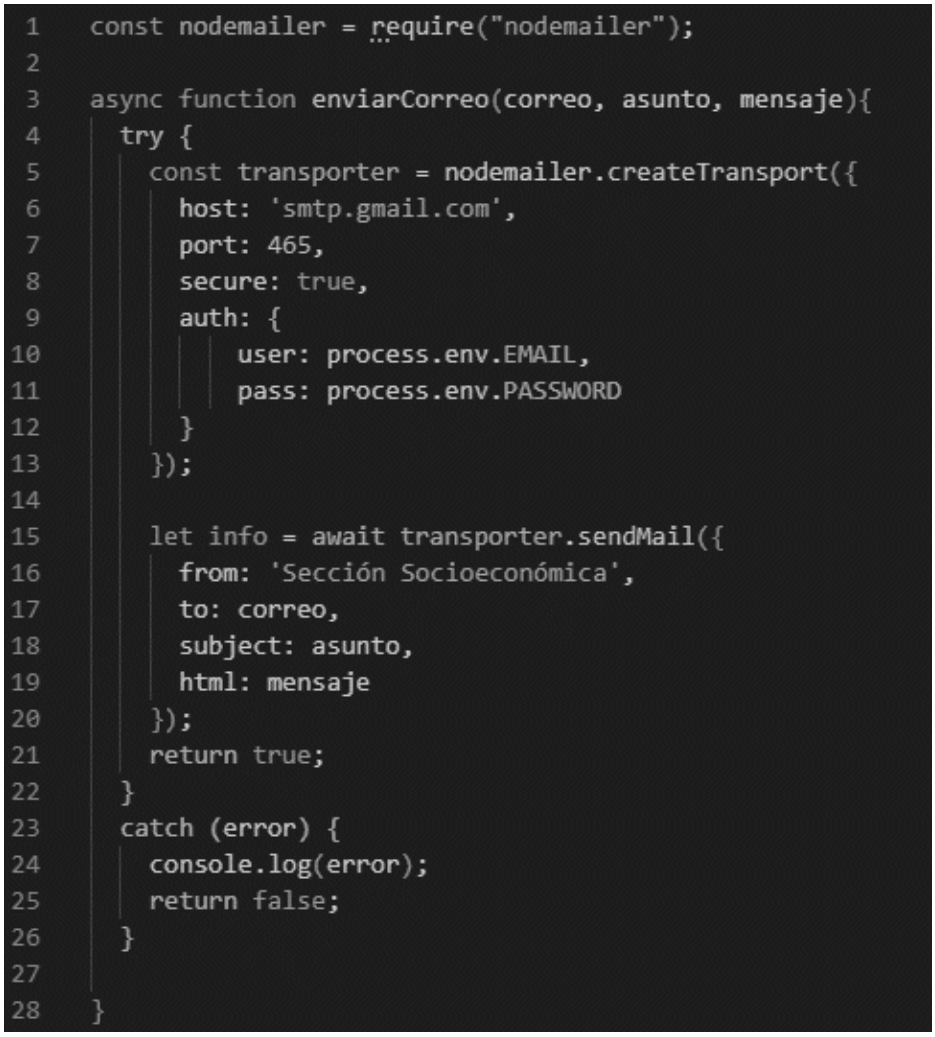
```
const multer = require('multer');

const almacenamiento = multer.diskStorage({
    destination: (req, file, cb) => {
        cb(null,'./storage_estudiantes');
    },
    filename: (req, file, cb) => {
        cb(null, file.originalname);
    }
});


const uploadMiddelware = multer({ storage: almacenamiento});
```

Fuente: elaboración propia, realizado con Visual Studio Code.

2.19.1.5.6. Nodemailer

Paquete de Node.js que se utiliza para el envío de notificaciones por correo electrónico, que se caracteriza por no utilizar dependencias y su fácil implementación. Su implementación requiere de su importación y posteriormente se realiza la configuración como el host, puerto, usuario, contraseña, entre otros. (Nodemailer, s.f.)

Figura 16. **Implementación nodemailer**



```
const nodemailer = require("nodemailer");

async function enviarCorreo(correo, asunto, mensaje){
  try {
    const transporter = nodemailer.createTransport({
      host: 'smtp.gmail.com',
      port: 465,
      secure: true,
      auth: {
          user: process.env.EMAIL,
          pass: process.env.PASSWORD
      }
    });

    let info = await transporter.sendMail({
      from: 'Sección Socioeconómica',
      to: correo,
      subject: asunto,
      html: mensaje
    });
    return true;
  }
  catch (error) {
    console.log(error);
    return false;
  }

}
```

Fuente: elaboración propia, realizado con Visual Studio Code.

2.19.1.5.7. PDF Merger JS

Paquete de Node.js que se utiliza para unir varios documentos PDF, ya sea completos o algunas páginas, y crea un nuevo documento. Este paquete se implementó para realizar la unificación de todos los documentos solicitados al estudiante y los informes que elaboraron los trabajadores sociales.

Figura 17. **Implementación pdf-merger-js**

```
let merger = new PDFMerger();
const unificado = i[0].concat(dr[0], dp[0]);

for(let documento of unificado){
    merger.add(documento.path_archivo);
}
await merger.save(file);
```

Fuente: elaboración propia, realizado con Visual Studio Code.

2.19.1.5.8. XLSX

Paquete de Node.js que se utiliza para crear archivos de Excel, este se utilizó para generar el reporte de todas las solicitudes que han sido denegadas.

Figura 18. **Implementación xlsx**

```
let newWB = xlsx.utils.book_new();
const newWS = xlsx.utils.json_to_sheet(resultado[0], { origin: 'A3'});
const h = [
    ['Reporte', date]
];
xlsx.utils.sheet_add_aoa(newWS, h, { origin: 'A1' });
xlsx.utils.book_append_sheet(newWB, newWS, "Denegadas");
const ruta = "./storage_reportes/Reporte.xlsx";
await xlsx.writeFile(newWB,ruta);
```

Fuente: elaboración propia, realizado con Visual Studio Code.

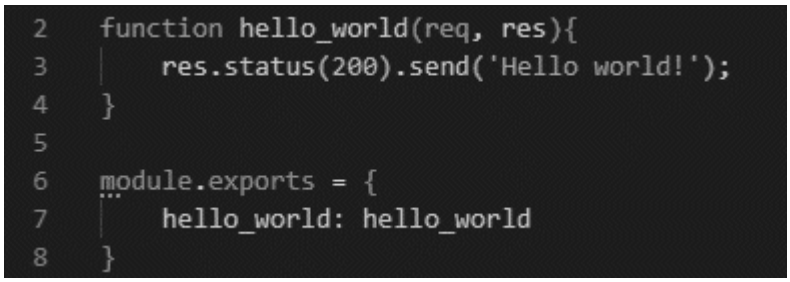
2.19.1.6. Config

Directorio que contiene los archivos de configuración del servidor, se hizo de esta manera para tener un orden establecido y no tener que repetir las configuraciones que se necesiten, sino que solo se requiere de su importación.

2.19.1.7. Controllers

Directorio que contiene todos los archivos con toda la lógica de la aplicación necesaria para responder las peticiones recibidas por las rutas, como buena práctica de programación es recomendable crear un controlador por cada ruta. Para llegar al controlador se debe pasar por ciertas validaciones que se explicarán en el middleware.

Figura 19. **Implementación de un controlador**

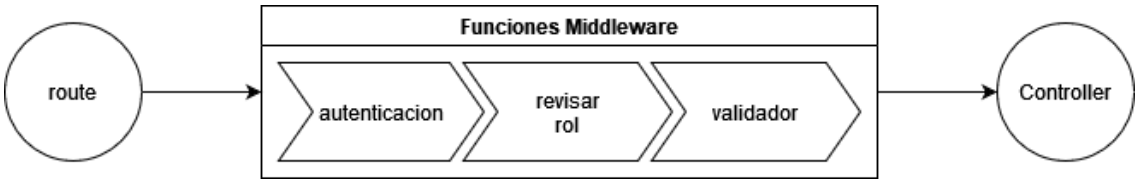


Fuente: elaboración propia, realizado con Visual Studio Code.

2.19.1.8. Middleware

Este directorio contiene todos archivos con las funciones intermedias que se utilizan para hacer las validaciones necesarias antes de llegar al controlador. Por ejemplo, una validación que se implementó fue la verificación del token, si este era válido seguía con las demás validaciones, en caso contrario se rechazaba la petición. En la siguiente imagen se muestra cómo se debe implementar los middlewares para realizar varias validaciones antes de llegar al controlador.

Figura 20. **Secuencia de un middleware**



Fuente: elaboración propia, realizado con draw.io.

2.19.1.9. Routes

En este directorio se almacenan todos los archivos que contienen los métodos de las rutas que utiliza el servidor, tales como POST, GET, PUT y DELETE. Cada ruta consta de tres partes:

- Endpoint, es el identificador que le permite al cliente apuntar a dicho servicio al momento de realizar una petición.

- Middlewares, funciones intermedias que son necesarias para realizar las validaciones por motivos de seguridad.

- Controlador, es la función que contiene la lógica de la petición que deberá responder al usuario que la solicitó.

Figura 21. **Estructura de una ruta**

```
router.get('/historial', autenticacionMiddleware, revisarRol(["estudiante"]), validadorEstudiante, get_historial);
router.get('/obtener-estados', autenticacionMiddleware, revisarRol(["estudiante"]), validadorEstudiante, obtener_estados);
```

Fuente: elaboración propia, realizado con Visual Studio Code.

2.19.1.10. Utils

Esta carpeta contiene los archivos con las funciones principales de los paquetes de Node.js que se emplearon en el proyecto tales como nodemailer, multer, jwt, entre otros.

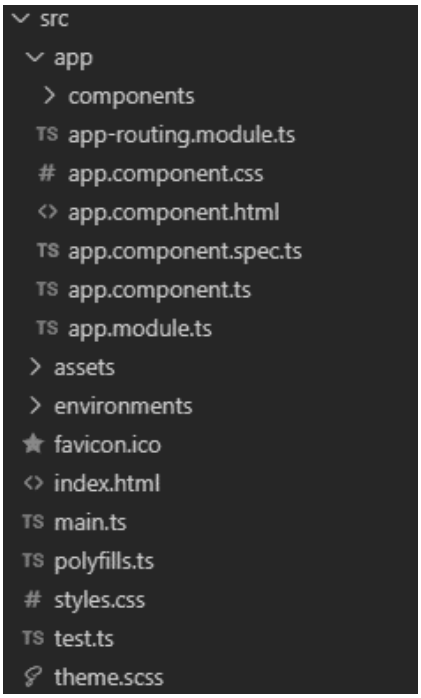
2.19.1.11. Validators

Esta carpeta contiene los archivos con las validaciones que se utilizaron para verificar que los parámetros de las solicitudes sean los solicitados por motivos de seguridad, utilizando express validator.

2.19.2. Cliente del proyecto

La aplicación web se desarrolló con el cliente de línea de comandos de Angular, con su versión estable al inicio del proyecto. De la misma manera que el servidor maneja una estructura de carpetas, Angular tiene su propia estructura que separa de manera ordenada todos sus archivos de configuración, dependencias y los archivos de desarrollo. La estructura de carpetas del cliente es más extensa a comparación del servidor, por tal razón solo se describirán los elementos más relevantes del proyecto.

Figura 22. **Estructura de carpetas del cliente**



Fuente: elaboración propia, realizado con Visual Studio Code.

Como se observa en la figura 22, el directorio src es el que contiene todos los archivos de desarrollo, tales como los estilos globales de la aplicación, la página de inicio, los módulos, servicios, interceptores, guardianes o guards, interfaces y los componentes por rol de cada usuario.

2.19.2.1. Componente

Angular trabaja con componentes, estos son los elementos principales de la aplicación, pueden generarse manualmente o utilizar los comandos del CLI para generarlos automáticamente. Un componente tiene los siguientes elementos:

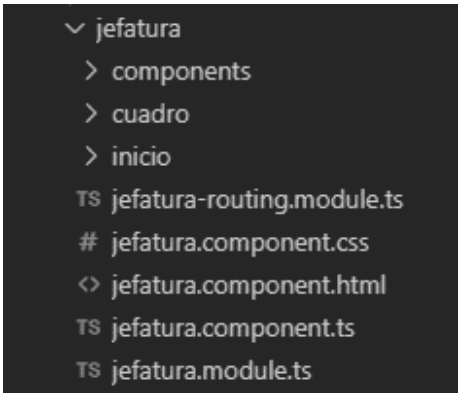
- Archivo HTML.

- Archivo de estilo que no tiene restricción del tipo que se desee implementar.

- Archivo de TypeScript, que contiene toda la lógica del componente.

- Archivo de especificación de componente, el cual puede ser omitido.

Figura 23. **Estructura de un componente**



Fuente: elaboración propia, realizado con Visual Studio Code.

2.19.2.1.1. Directivas

Las directivas son clases que tienen el objetivo de agregar funcionalidades adicionales a los elementos que se utilizan en Angular. Existen varios tipos de directivas, las más importantes que se utilizaron en el proyecto son las directivas estructurales incorporadas que son utilizadas para manipular la estructura del diseño HTML, las que se utilizaron con mayor frecuencia se describen a continuación:

- NgIf, esta directiva crea o elimina condicionales de las subvistas la de la plantilla.

Figura 24. **Implementación NgIf**

```
<div *ngIf="interrogante">
    <mat-icon class="icono3">help_outline</mat-icon>
    <h3><b>¿Desea enviar la documentación requerida?</b></h3>
    <p>Antes de confirmar debe verificar que todos los documentos se hayan seleccionado correctamente.</p>
</div>
```

Fuente: elaboración propia, realizado con Visual Studio Code.

- NgFor, esta directiva repite un nodo para cada elemento de una lista.

Figura 25. **Implementación NgFor**

```
<tr *ngFor="let documento of lista; index as i;" class="border_bottom">
    <td class="tdDescripcion">···
    </td>
    <td style="text-align: center;" class="alternative">···
    </td>
    <td class="tdTexto">···
    </td>
    <td>···
    </td>
</tr>
```

Fuente: elaboración propia, realizado con Visual Studio Code.

2.19.2.1.2. Servicios

Los servicios en angular son clases que tienen un conjunto de instrucciones o funciones específicas que son instanciables, reutilizables y disponibles en todo tiempo para los componentes que utiliza la aplicación. Estas funciones se pueden implementar directamente desde los componentes, pero no

se recomienda ya que es considerado como una mala práctica de programación. La mayor utilidad que se le dio en este proyecto fue para la comunicación con el servidor, en acciones como agregar, obtener, actualizar y eliminar información.

Figura 26. **Implementación services**

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { Beca } from '../interfaces/beca';

@Injectable({
  providedIn: 'root'
})
export class BecasService {

  constructor(private http: HttpClient) { }

  getDescripcion(): Observable<Beca[]>{
    return this.http.get<Beca[]>('./assets/data/tipos.json');
  }


  getEstados(): Observable<Beca[]>{
    return this.http.get<Beca[]>('./assets/data/estados.json');
  }

}
```

Fuente: elaboración propia, realizado con Visual Studio Code.

2.19.2.1.3. Guards

Guards o guardianes son interfaces proporcionadas por Angular que tiene como función principal el control del acceso a una determinada ruta, que se

implementó en determinada clase. Existen 4 tipos de guardianes donde se pueden implementar:

- *CanActivate*: este tipo es el más comúnmente utilizado para activar una ruta.

- *CanActivateChild*: este tipo controla si se pueden activar los componentes hijos de una ruta.

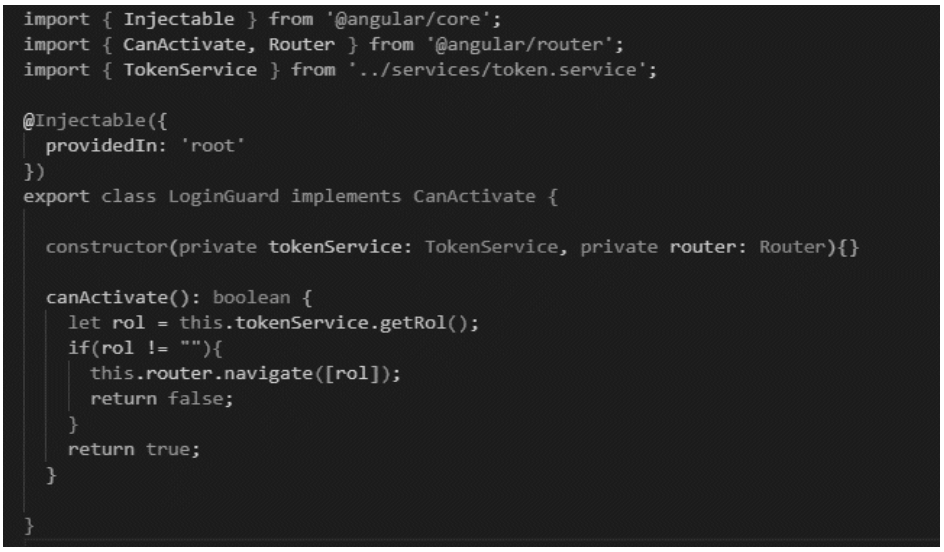
- *CanLoad*: este tipo controla si una ruta puede cargarse o no, muy útil para los módulos que tengan carga diferida.

- *CanDeactivate*: este tipo controla si el usuario puede salir de una ruta.

Por motivos de seguridad los guardianes se implementaron en los siguientes casos:

- Inicio de sesión, cuando el usuario haya iniciado sesión automáticamente desde Angular se almacenarán en cookies el token de sesión y el refreshtoken, si este decidiera regresar al inicio de sesión se activará el guardián y comprobará, por medio de un servicio, que el rol esté almacenado en el cliente, si el rol no existe entonces será dirigido a la ventana de inicio de sesión, en caso contrario será dirigido a la ventana principal del rol que le corresponde.

Figura 27. **Guardian de inicio de sesión**



```
import { Injectable } from '@angular/core';
import { CanActivate, Router } from '@angular/router';
import { TokenService } from '../services/token.service';

@Injectable({
  providedIn: 'root'
})
export class LoginGuard implements CanActivate {

  constructor(private tokenService: TokenService, private router: Router){}

  canActivate(): boolean {
    let rol = this.tokenService.getRol();
    if(rol != ""){
      this.router.navigate([rol]);
      return false;
    }
    return true;
  }

}
```

Fuente: elaboración propia, realizado con Visual Studio Code.

- La implementación del guardián se realiza en el módulo de rutas, se debe importar y en el apartado de rutas se debe seleccionar el tipo de guardián.

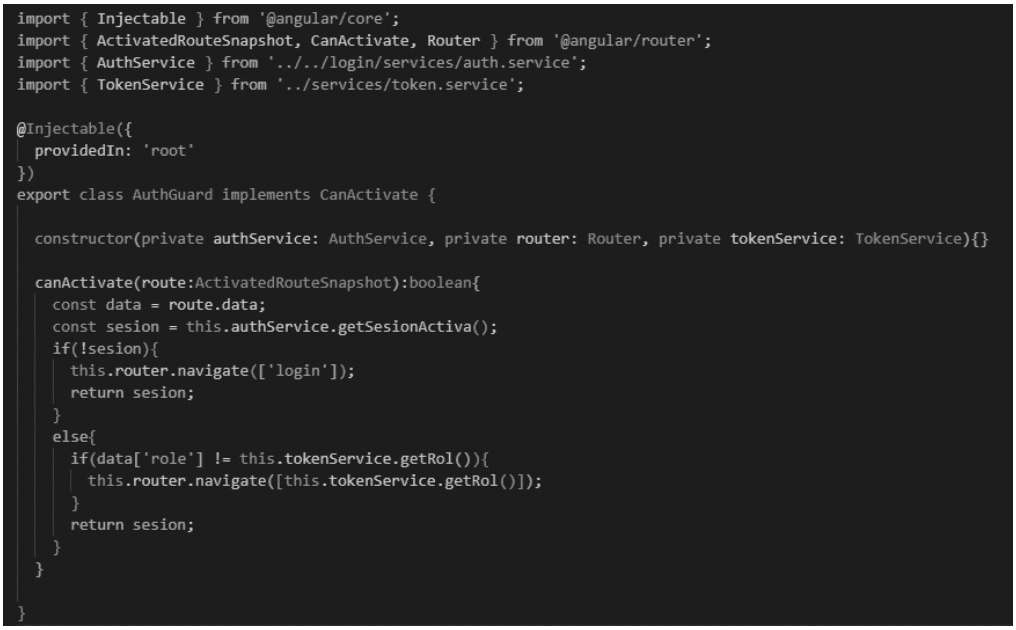
Figura 28. **Implementación del guardián de inicio de sesión**

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { LoginComponent } from './components/login/login.component';
import { AuthGuard } from './components/shared/guards/auth.guard';
import { LoginGuard } from './components/shared/guards/login.guard';

const routes: Routes = [
  {
    path: '', redirectTo: 'login', pathMatch: 'full'
  },
  {
    path: 'login',
    component: LoginComponent,
    canActivate: [LoginGuard]
  },
```

Fuente: elaboración propia, realizado con Visual Studio Code.

- Autenticación de roles, cuando el usuario haya iniciado sesión automáticamente desde Angular se almacenarán en cookies el token de sesión y el refreshtoken, cuando el usuario desee ingresar a cualquiera de las rutas disponibles al momento de hacer clic se activará el guardián verificará la existencia del token de sesión, si el token no existe entonces será dirigido al inicio de sesión en caso contrario se comprueba el rol del usuario, si el rol del usuario no está autorizado para acceder a la ruta será dirigido a la ventana principal que le corresponde, en caso contrario tendrá acceso a la ruta deseada.

Figura 29. **Guardián de autenticación de roles**



```
import { Injectable } from '@angular/core';
import { ActivatedRouteSnapshot, CanActivate, Router } from '@angular/router';
import { AuthService } from '../../login/services/auth.service';
import { TokenService } from '../services/token.service';

@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {

  constructor(private authService: AuthService, private router: Router, private tokenService: TokenService){}

  canActivate(route:ActivatedRouteSnapshot):boolean{
    const data = route.data;
    const sesion = this.authService.getSesionActiva();
    if(!sesion){
      this.router.navigate(['login']);
      return sesion;
    }
    else{
      if(data['role'] != this.tokenService.getRol()){
        this.router.navigate([this.tokenService.getRol()]);
      }
      return sesion;
    }
  }

}
```

Fuente: elaboración propia, realizado con Visual Studio Code.

- La implementación es la misma que en el caso anterior.

- Expiración de sesión, El guardián en este caso se encarga de verificar la caducidad del token, como buena práctica de programación se sugiere darle una caducidad menor al token de sesión y una caducidad mayor al refreshtoken de tal manera que este pueda consultar al servidor y generar nuevos tokens, de esta forma se evita que el usuario este iniciando sesión en repetidas ocasiones.

Figura 30. **Guardian de expiración de sesión**

```
import { Injectable } from '@angular/core';
import { CanActivateChild } from '@angular/router';
import { TokenService } from '../services/token.service';

@Injectable({
  providedIn: 'root'
})
export class TokenGuard implements CanActivateChild {

  constructor(private tokenService: TokenService){}

  canActivateChild(): Promise<boolean>{
    return this.tokenService.checkToken();
  }
}
```

Fuente: elaboración propia, realizado con Visual Studio Code.

- La implementación se realizó en el módulo de rutas del componente de cada rol.

Figura 31. **Implementación del guardián de expiración de sesión**

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { TokenGuard } from '../shared/guards/token.guard';
import { CuadroComponent } from './cuadro/cuadro.component';
import { InicioComponent } from './inicio/inicio.component';
import { JefaturaComponent } from './jefatura.component';

const routes: Routes = [
  {
    path: '', component: JefaturaComponent, children:[
      {
        path: '', component: InicioComponent
      },
      {
        path: 'cuadro', component: CuadroComponent
      }
    ],
    canActivateChild: [TokenGuard]
  }
];
```

Fuente: elaboración propia, realizado con Visual Studio Code.

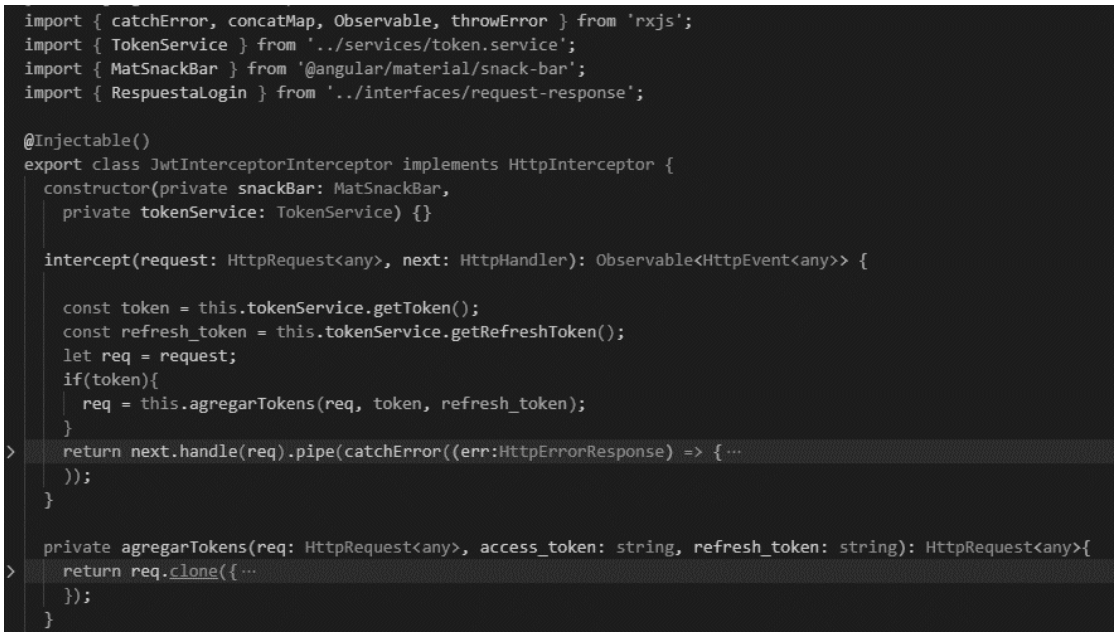
2.19.2.1.4. Interceptor

Es un servicio de Angular que tiene como función principal interceptar las solicitudes HTTP entrantes o salientes. Esta funcionalidad es de gran uso, específicamente en la reutilización, en el proyecto se utilizó para los siguientes casos.

- Solicitudes salientes, cuando se realiza una petición se debe adjuntar en el encabezado los tokens, para evitar escribir las mismas instrucciones en todas las solicitudes se utilizó el interceptor para que agregue de forma automática los tokens.

- Solicitudes entrantes, cuando una petición se ha rechazado se debe verificar las razones por las cuales el servidor dio ese resultado y determinar si es posible realizar la petición para actualizar los tokens.

Figura 32. **Interceptor**

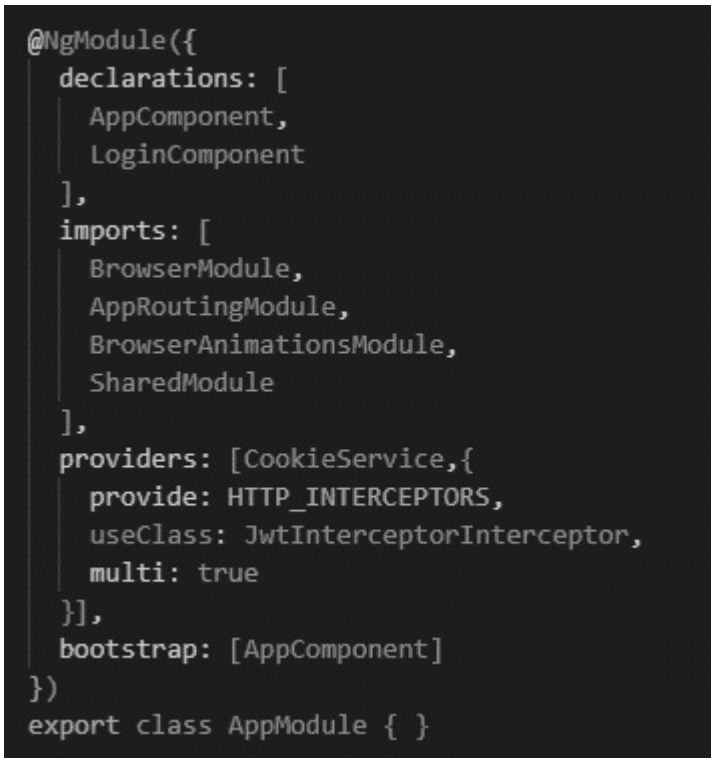


```
import { catchError, concatMap, Observable, throwError } from 'rxjs';
import { TokenService } from '../services/token.service';
import { MatSnackBar } from '@angular/material/snack-bar';
import { RespuestaLogin } from '../interfaces/request-response';

@Injectable()
export class JwtInterceptorInterceptor implements HttpInterceptor {
  constructor(private snackBar: MatSnackBar,
    private tokenService: TokenService) {}

  intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {

    const token = this.tokenService.getToken();
    const refresh_token = this.tokenService.getRefreshToken();
    let req = request;
    if(token){
      req = this.agregarTokens(req, token, refresh_token);
    }
    return next.handle(req).pipe(catchError((err:HttpErrorResponse) => {···
    ));
  }

  private agregarTokens(req: HttpRequest<any>, access_token: string, refresh_token: string): HttpRequest<any>{
    return req.clone({···
    });
  }
```

Fuente: elaboración propia, realizado con Visual Studio Code.

La implementación del interceptor creado se debe agregar en el módulo principal de la aplicación. Como se puede observar en la siguiente imagen, el interceptor se debe agregar en el apartado de secciones, el parámetro multi permite que se puedan agregar varios interceptores.

Figura 33. **Implementación del interceptor**



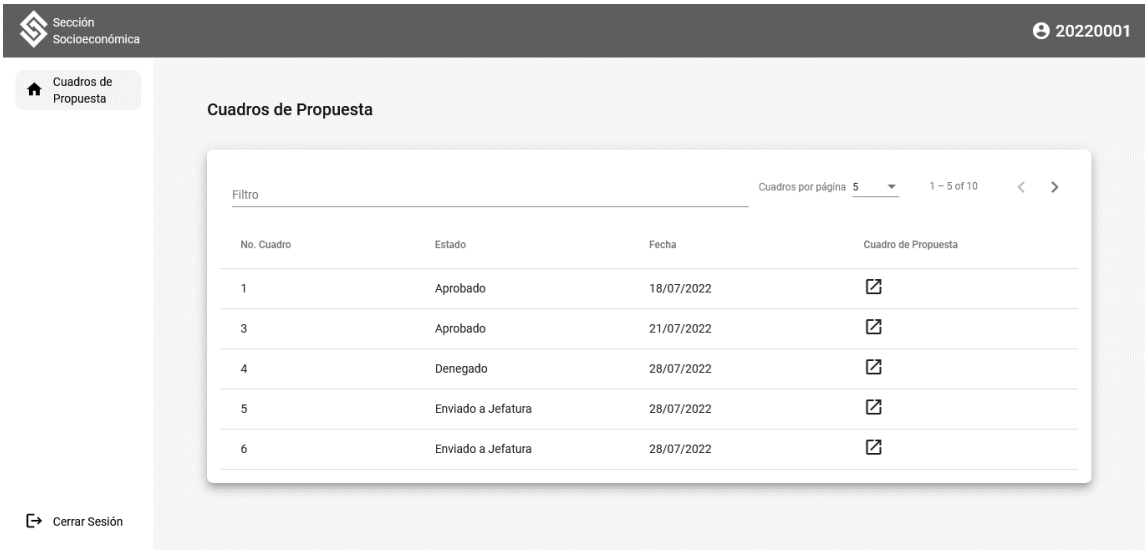
```
@NgModule({
  declarations: [
    AppComponent,
    LoginComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    SharedModule
  ],
  providers: [CookieService,{
    provide: HTTP_INTERCEPTORS,
    useClass: JwtInterceptorInterceptor,
    multi: true
  }],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Fuente: elaboración propia, realizado con Visual Studio Code.

2.19.2.1.5. Diseño

El diseño de la aplicación se realizó con Angular Material, gracias a esta biblioteca el diseño de las barras laterales, tablas, filtros, cajas de texto, botones, entre otros elementos, son agradables a la vista del usuario y su experiencia es mejor. En la siguiente imagen se muestra el diseño desde el punto de vista desde el rol de jefatura, se puede apreciar la barra lateral con todas las opciones que tiene disponibles, una tabla con todos los cuadros de propuesta que la secretaria ha enviado, un cuadro de texto que filtra la información de la tabla al ingresar caracteres, una cantidad determinada de elementos que se puede modificar por medio de una opción y dos botones para navegar entre páginas de la tabla.

Figura 34. **Diseño menú jefatura**



Fuente: elaboración propia, realizado con Visual Studio Code.

Una de las principales ventajas de haber desarrollado el proyecto con Angular y Angular material es el uso de los componentes y su reutilización, en el caso del formulario de documentación requerida y documentación posterior se debe mostrar la descripción del documento solicitado, el estado, un botón de carga de archivos, un botón para quitar el archivo seleccionado y por último un botón para subir todos los documentos.

Existen seis tipos de becas, cada uno con sus respectivos documentos que la Sección Socioeconómica solicita, desarrollar un formulario por cada tipo de beca no era una solución óptima ya no se hubiera aprovechado al máximo el uso de las herramientas seleccionadas, por tal razón el uso de directivas y componentes solucionó esta situación. Por medio de la directiva NgFor se fueron creando los elementos descritos con anterioridad y por medio de los componentes hijos se logró realizar el formulario de forma dinámica, ya que

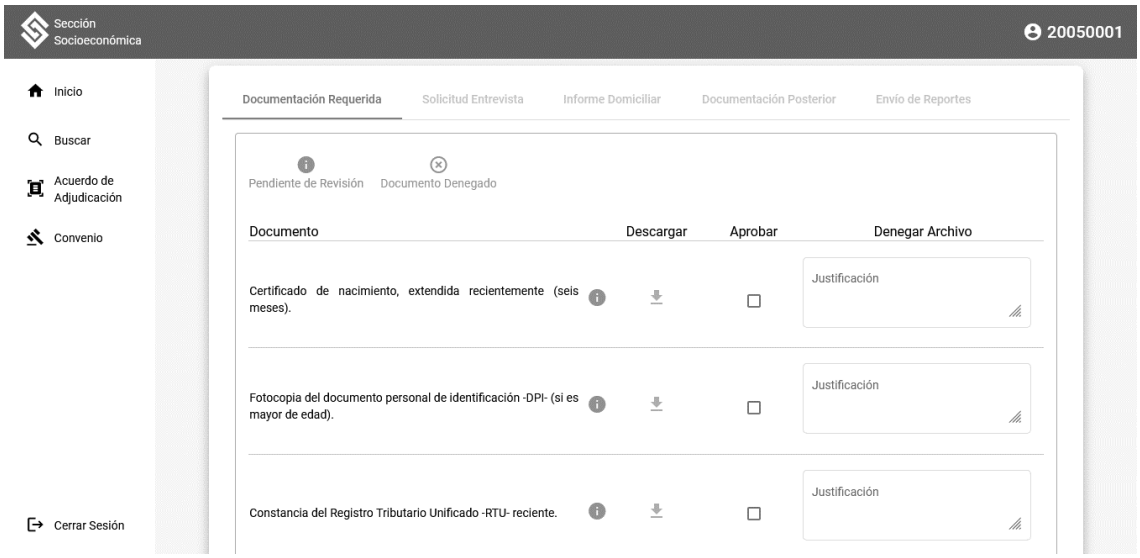
gracias a Angular se puede realizar la comunicación desde un componente padre hacia sus componentes hijos y viceversa, de esta manera la implementación se realizó para obtener los archivos seleccionados.

Figura 35. **Diseño formulario documentación requerida**

Fuente: elaboración propia, realizado con Visual Studio Code.

La gestión de la documentación es otra situación que se solucionó de la misma manera fue con la gestión de la documentación requerida y posterior, el trabajador social tenía que observar la descripción del documento a revisar, una opción para descarga el documento, una casilla para aprobar el documento, una caja de texto para agregar una justificación en caso de que el documento se denegara, y un botón para guardar el estado de todos los documentos. Hacerlo de una forma estática no era una solución óptima para esta situación por la cantidad de tipos de becas, por tal razón se utilizó la directiva NgFor para crear cada elemento descrito anteriormente y la comunicación entre el componente padre hacia los hijos y viceversa, se logró obtener el estado de todos los documentos para posteriormente enviarlos en una petición al servidor.

Figura 36. **Formulario gestión de documentación requerida**



Fuente: elaboración propia, realizado con Visual Studio Code.

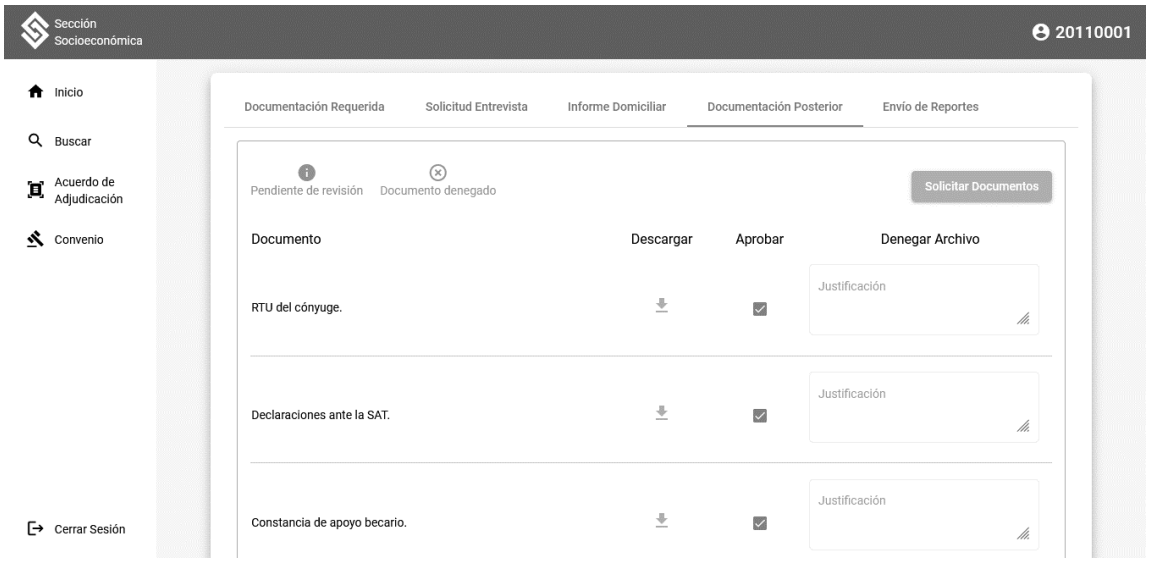
La solicitud de documentación posterior es otro caso que se solucionó de la misma manera ya que existe un listado de los posibles documentos que se pueden solicitar a los estudiantes y los trabajadores sociales deben seleccionar, utilizando la directiva NgFor se crearon todos los posibles documentos junto con una casilla para seleccionar los documentos, y que por medio de la comunicación entre componentes se logró obtener el listado de documentos marcados por el trabajador social.

El siguiente reto era crear otro formulario dinámico para que el estudiante pueda subir los documentos solicitados y el trabajador social pueda realizar la gestión. En este caso no había una solución estática para los formularios ya que existiría un gran número posibles documentos solicitados por tal razón se ve como única opción la utilización de las directivas y la comunicación entre componentes.

Figura 37. **Formulario solicitud de documentos posteriores**

Fuente: elaboración propia, realizado con Visual Studio Code.

Figura 38. **Formulario gestión documentación posterior**



Fuente: elaboración propia, realizado con Visual Studio Code.

2.20. Ambiente desarrollo de la aplicación

La aplicación se desarrolló con los siguientes recursos:

- HP Envy dv6, Intel Core i7, 8 GB de memoria RAM, 480 GB de disco duro, Sistema Operativo Windows 10 Pro.
- Node.js v16.13.1
- Angular 13.1.0
- MySQL 8.0.28

2.21. Costos del proyecto

Para el cumplimiento y realización de este proyecto es de suma importancia tener recursos económicos, estos se describen en la siguiente tabla.

Tabla III. **Costo del proyecto**

Recursos	**Cantidad**	**Costo Unitario**	**Subtotal**
Desarrollador	6 meses (4 horas diarias)	Q. 6,300.00/mes	Q. 37,800.00
Asesoramiento	1 año (1 hora por semana)	Q. 975.00/semana	Q. 50,700.00
Servicio energía eléctrica	6 meses	Q. 300.00/mes	Q. 1800.00
Servicio de internet	6 meses	Q. 300.00/mes	Q. 1800.00
Mobiliario y equipo	1	Q. 1,500.00	Q. 1,500.00
Computadora personal	1	Q. 6,500.00	Q. 6,500.00
Mantenimiento y limpieza computadora personal	6 meses	Q. 200.00/mes	Q. 1,200.00

Continuación de la tabla III.

Total	Q. 101,300.00

Fuente: elaboración propia.

2.22. Beneficios del proyecto

Los beneficios que la Sección Socioeconómica de la Universidad de San Carlos de Guatemala obtuvieron se listan a continuación:

- Facilitar el proceso de envío de documentos a los estudiantes que solicitan beca.
- Seguimiento del proceso de la solicitud por parte de los estudiantes.
- Orden y simplicidad para los trabajadores sociales durante el proceso de revisión de los documentos recibidos por solicitud.
- Mejor control de envío de informes de diagnóstico, domiciliar y expedientes completos.
- Mejorar la gestión de envío de cuadros de propuesta entre la secretaria de comisión y jefatura.
- Facilidad para la gestión de archivos como convenios y acuerdos de adjudicación.
- Agilización para la gestión de trabajadores sociales y usuarios del sistema.

- Obtención de reportes de solicitudes denegadas.

- Reducción de tiempo durante los primeros pasos de las solicitudes de becas.

- Automatizar el proceso de notificaciones por medio de correos electrónicos en determinados puntos del proceso de solicitud de beca.

- Escalabilidad del módulo de solicitudes de becas para agregar más funcionalidades en el futuro.

3. FASE ENSEÑANZA APRENDIZAJE

3.1. Capacitación propuesta

La capacitación del nuevo sistema que obtuvo la Sección Socioeconómica se llevó a cabo de la siguiente manera:

3.1.1. Capacitación a usuario

Para los usuarios jefatura, secretaria de comisión, trabajador social, secretaria de recepción y secretaria de convenios de la Sección Socioeconómica se realizó la capacitación por medio de reuniones virtuales de tal manera que todos los involucrados conocieran a detalle el sistema y las opciones disponibles que cada uno tiene disponible. Incluso este tipo de capacitaciones fue de gran ayuda para corregir ciertos aspectos del proceso de solicitud de becas y que la experiencia de usuario mejorara. En el caso de los estudiantes únicamente se cuenta con el manual de usuario que se describe en la siguiente sección.

3.2. Material Elaborado

En la fase de enseñanza tanto para los colaboradores de la Sección Socioeconómica como de los colaboradores en el mantenimiento del proyecto requieren de un material visual para el uso correcto y mantenimiento, por tal razón era necesario tener dos categorías para los manuales.

3.2.1. Manual de usuario

Se realizaron manuales de usuario para cada rol que utiliza el sistema, cada documento contiene la descripción detallada de todas las funcionalidades disponibles que le corresponde a cada rol, de tal manera que de existir alguna duda o el usuario desee prevenir posibles errores que se puede cometer durante el uso del sistema, cada documento le sea de gran ayuda para solventar todas las dudas posibles.

3.2.2. Manual técnico

Este documento contiene la descripción detalla de todas las características técnicas principales del desarrollo del backend y frontend, tales como teoría de las herramientas utilizadas, las clases que se desarrollaron, funciones principales, patrones de diseño, diagrama entidad relación y arquitectura. Este documento tiene como objetivo principal proveer la información suficiente para que en su debido momento se pueda realizar mantenimiento o se utilice como base para realizar cambios o agregar funcionalidades a cada uno de los módulos desarrollados.

CONCLUSIONES

1. Los estudiantes que han solicitado beca ahora podrán conocer el estado del proceso que conlleva su solicitud en tiempo real y en cualquier momento, sin necesidad de consultar directamente con la Sección Socioeconómica. Además, el envío de los documentos solicitados le será más fácil y podrán conocer el estado de los documentos si han sido aprobados o denegados.

2. El módulo para los trabajadores sociales les permite llevar un control ordenado de todas las solicitudes que se les ha asignado y podrán realizar la gestión de toda la documentación que requiere el proceso de solicitud de beca. Además, podrán obtener todos los acuerdos de adjudicación y convenios que sean cargados al sistema por la secretaria de comisión y secretaria de convenios.

3. Jefatura ahora tiene un módulo donde podrá revisar todos los cuadros de propuesta enviados por la secretaria de comisión y en caso de necesitar correcciones se podrán agregar las anotaciones necesarias.

4. La secretaria de comisión se ha beneficiado con su módulo ya que ahora puede obtener de manera ordenada toda la documentación por solicitud de beca, puede enviar a jefatura los cuadros de propuesta los acuerdos de adjudicación los podrá subir al sistema y asignarlos a los trabajadores sociales, podrá realizar la gestión de los trabajadores sociales, usuarios y documentación posterior, obtendrá el reporte de todas las solicitudes

denegadas por medio de un archivo de Excel y manejará el estado de la automatización de la obtención y denegación de solicitudes.

5. Se implementó un módulo para la secretaria de convenios de tal manera que ella pueda realizar la gestión de todos los convenios obtenidos en la adjudicación de becas y podrá enviarlos a cada trabajador social que le corresponda.

6. La secretaria de convenios será notificada cuando la documentación requerida se haya validado por los trabajadores sociales, podrá acceder a todas las solicitudes activas y realizar la gestión de todos los usuarios y trabajadores sociales del sistema.

7. Se realizó la capacitación a los usuarios de la Sección Socioeconómica, la cual proporcionará compresión y el uso correcto del sistema. Además, se cuenta con los manuales de usuario que les brinda una ayuda en caso de que ocurran dudas en el uso del sistema.

RECOMENDACIONES

1. Brindar constante soporte a la Sección Socioeconómica porque es una organización que vela por el bien del estudiante y requiere de un amplio esfuerzo de trabajo por parte de todo el personal que lo integra, y actualmente no se han suplido las necesidades que esta ha presentado con el paso del tiempo.

2. Determinar otros pasos dentro del proceso de solicitud de becas que se deben cubrir, para tomarlos como requerimientos y plantearlos como extensiones del sistema y que su resolución sea más fácil y rápido.

3. Establecer más medios de comunicación con el Departamento de Procesamiento de Datos para evitar demoras en las próximas extensiones del sistema o mantenimiento que este requiera.

4. Involucrar al personal de la Sección Socioeconómica para realizar capacitaciones a los estudiantes que utilizarán el sistema, con el objetivo de que estos tengan un uso correcto del sistema, así como retroalimentación para incorporar cambios o mejoras.

5. Proporcionar capacitaciones específicamente a los trabajadores sociales, que les brinde todos los parámetros y requerimientos correctos que deben revisar en la documentación que se solicita a los estudiantes, para evitar las constantes correcciones por parte de la secretaria de comisión y de esta manera agilizar el proceso de las solicitudes de becas.

REFERENCIAS

1. Angular (28 de febrero, 2022). *Angular*. [Mensaje en un blog] Recuperado de https://angular.io/guide/what-is-angular.

2. Angular Material (s.f.). *Getting started Angular Material*. [Mensaje en un blog] Recuperado de https://material.angular.io/guide/getting-started.

3. Express (2017). *Express Infraestructura de aplicaciones web Node.js* [Mensaje en un blog] Recuperado de https://expressjs.com/es.

4. Okta (s.f.). *Introduction to JSON Web Tokens* (s.f.). [Mensaje en un blog] Recuperado de https://jwt.io/introduction.

5. Node.js (s.f.). *Usage and Example Node.js v16.16.0 documentation*. [Mensaje en un blog] Recuperado de https://nodejs.org/dist/latest-v16.x/docs/api/synopsis.html.

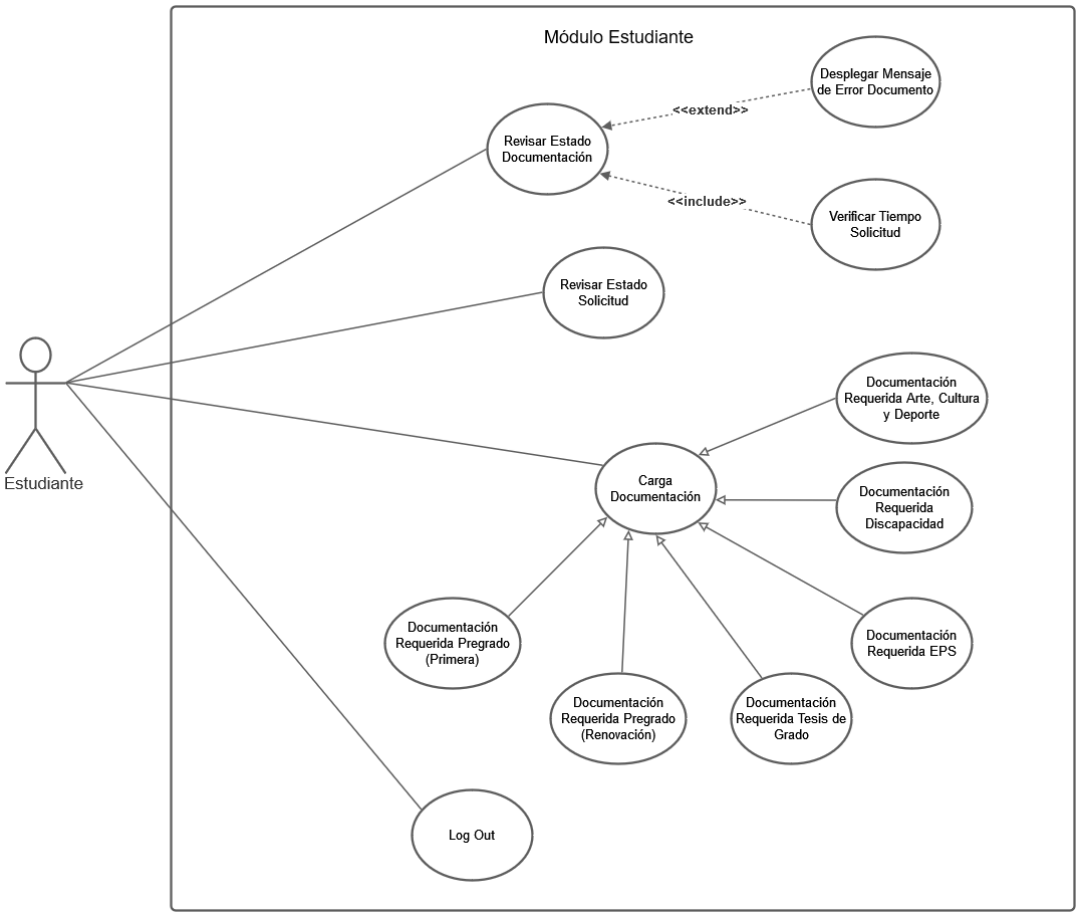
6. Nodemailer (s.f.). *Nodemailer*. [Mensaje en un blog] Recuperado de https://nodemailer.com/about.

7. NPM (s.f.). *pdf-merger-js.* [Mensaje en un blog] Recuperado de https://www.npmjs.com/package/pdf-merger-js.

8. USAC, Becas (10 de octubre, 1959). *Becas USAC.* [Mensaje en un blog] https://becas.usac.edu.gt/index.php/acerca-de-seccion-.socioeconomica/.

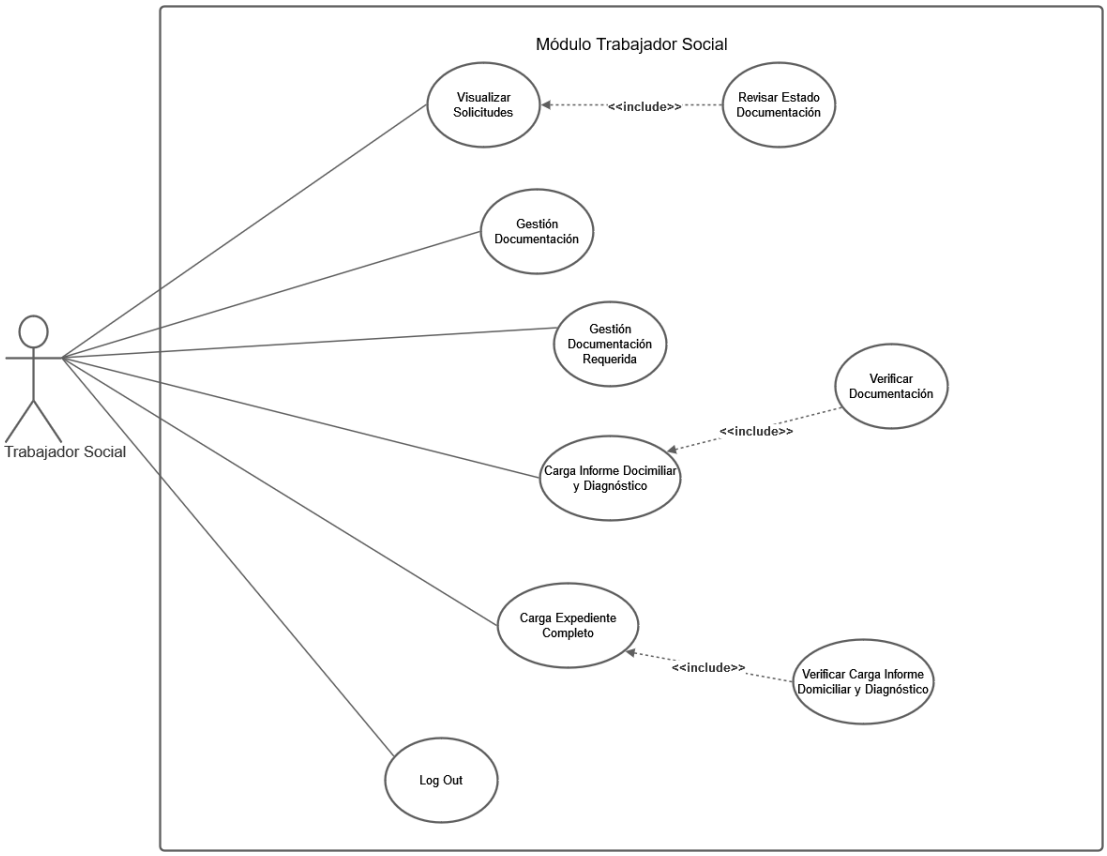
APÉNDICES

Apéndice 1. **Diagrama de caso de uso estudiante**



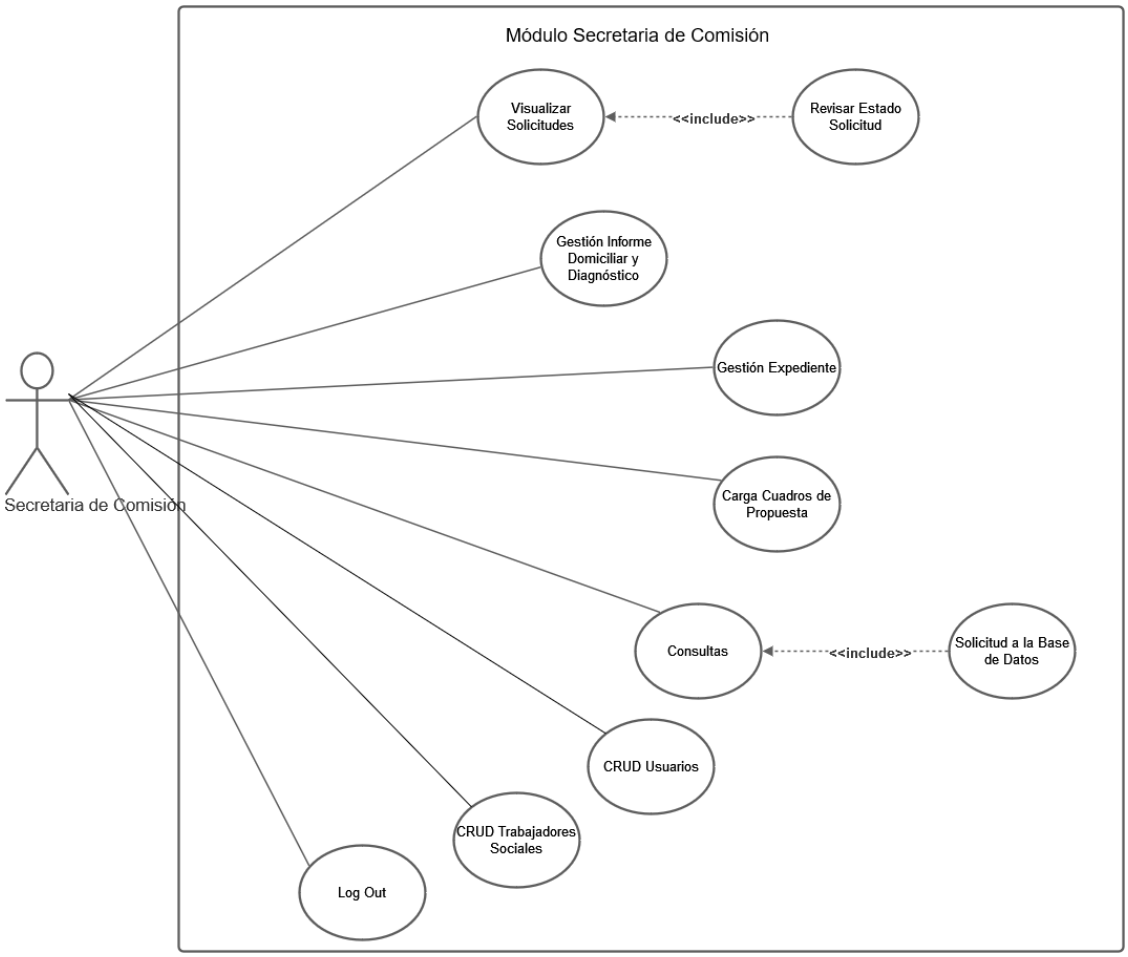
Fuente: elaboración propia, realizado con draw.io.

Apéndice 2. **Diagrama de caso de uso trabajador social**



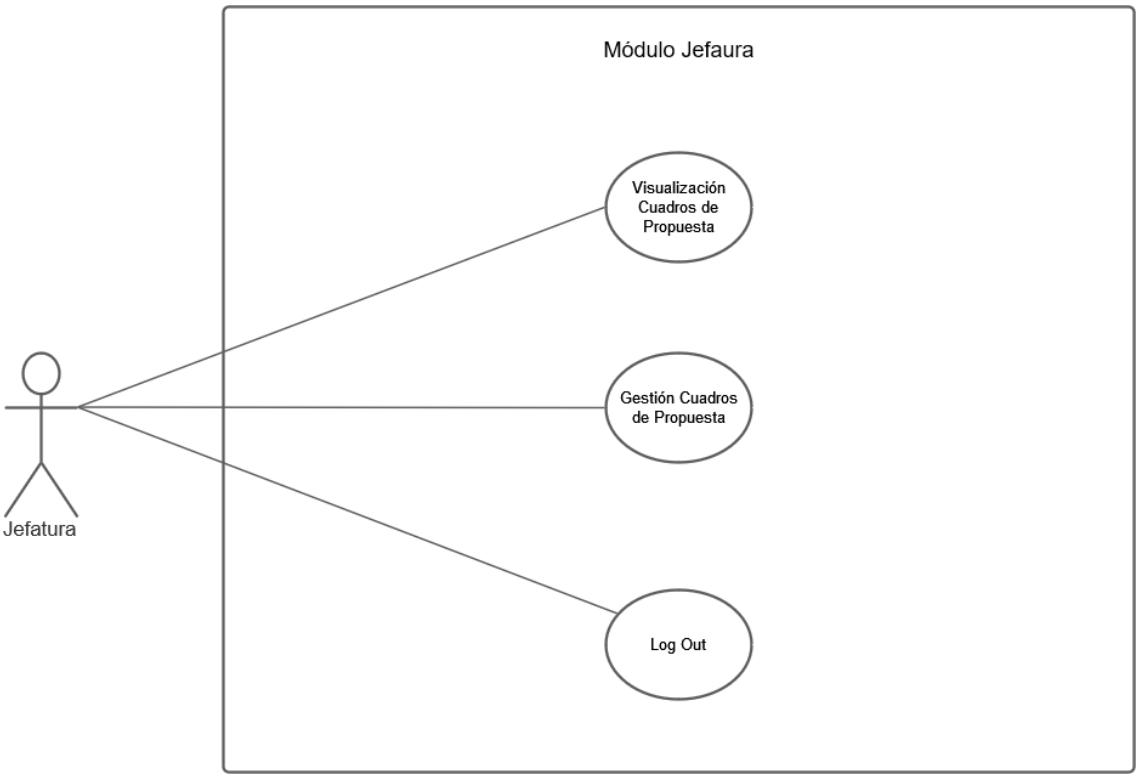
Fuente: elaboración propia, realizado con draw.io.

Apéndice 3. **Diagrama de caso de uso secretaria de comisión**



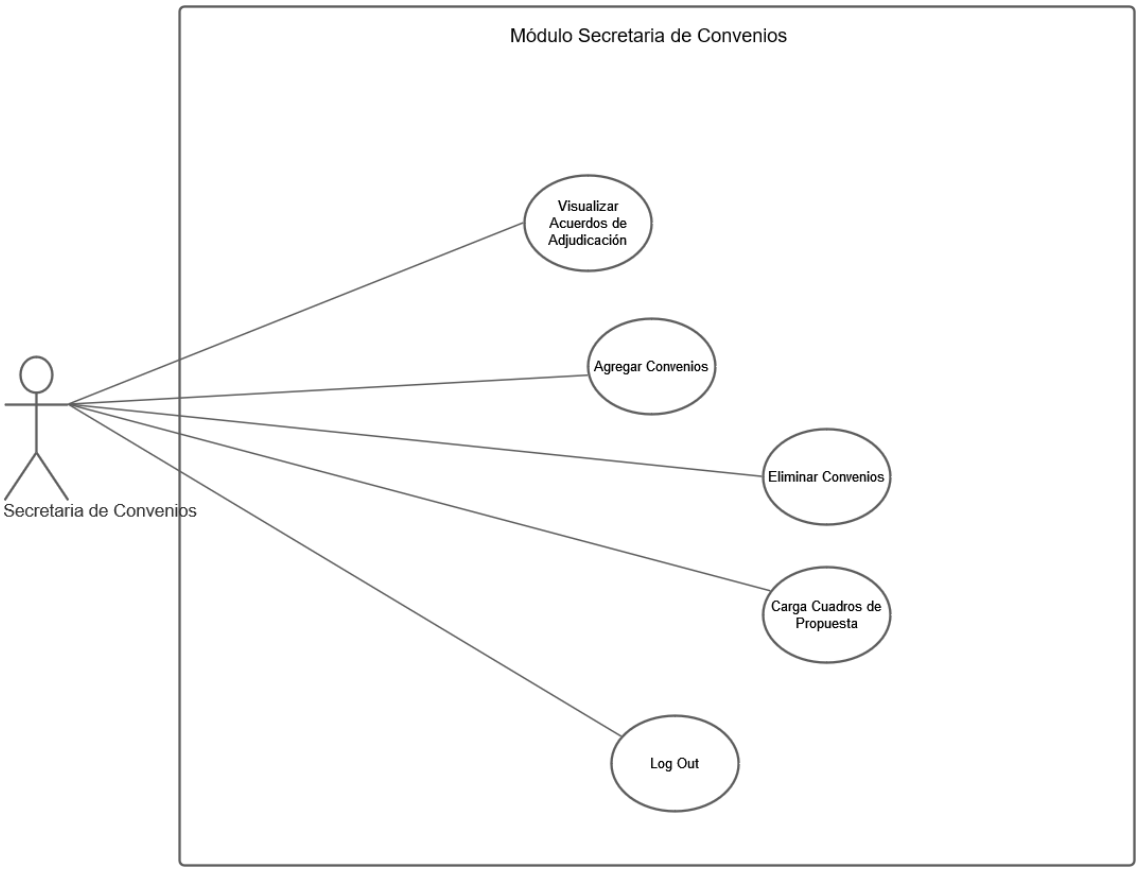
Fuente: elaboración propia, realizado con draw.io.

Apéndice 4. **Diagrama de caso de uso jefatura**



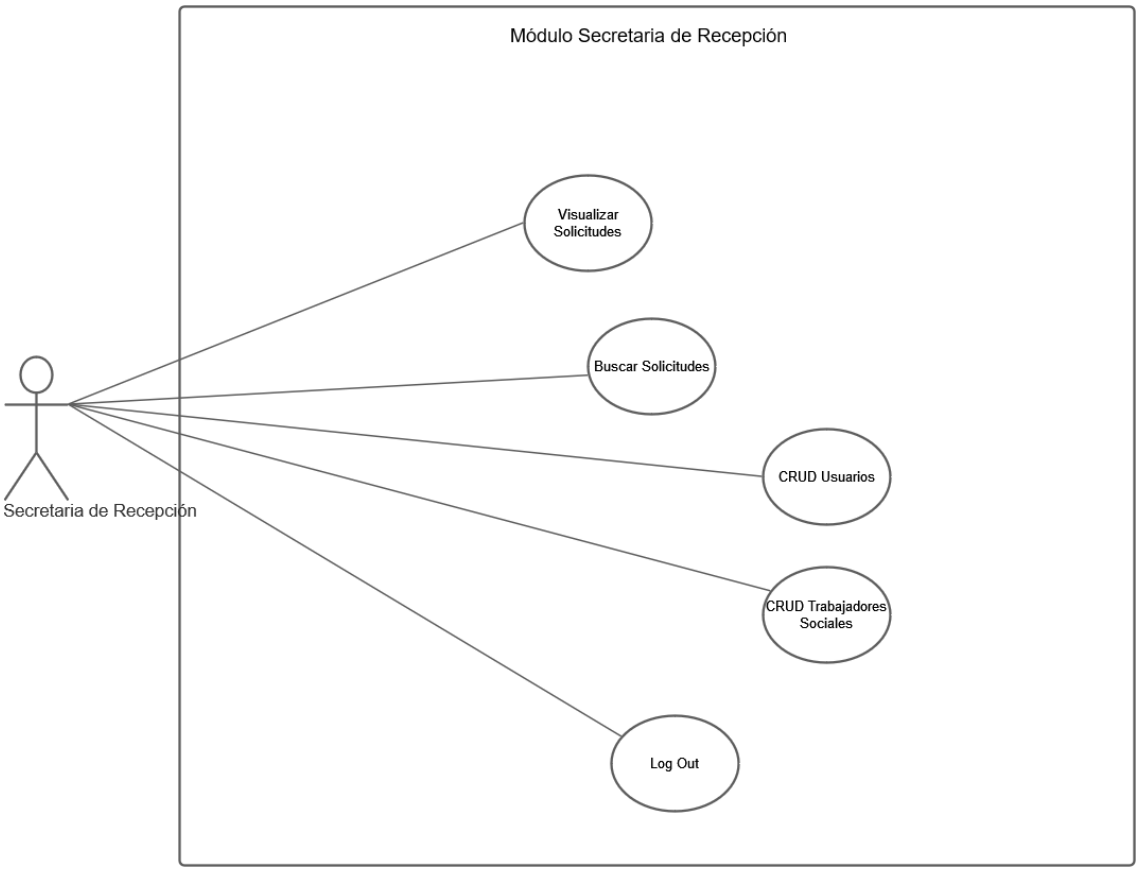
Fuente: elaboración propia, realizado con draw.io.

Apéndice 5. **Diagrama de caso de uso secretaria de convenios**



Fuente: elaboración propia, realizado con draw.io.

Apéndice 6. **Diagrama de caso de uso secretaria de recepción**



Fuente: elaboración propia, realizado con draw.io.