**Universidad Nacional Mayor de San Marcos**
**Universidad del Perú. Decana de América**
Facultad de Ingeniería de Sistemas e Informática
Escuela Profesional de Ingeniería de Sistemas

# Implementación de una arquitectura micro-frontend para optimizar el desarrollo y despliegue de la aplicación web del sistema de información universitaria de la SUNEDU

**TRABAJO DE SUFICIENCIA PROFESIONAL**

Para optar el Título Profesional de Ingeniero de Sistemas

**AUTOR**

Mario GARCIA CHIHUANHUAYLLA

**ASESOR**

Dr. Carlos Edmundo NAVARRO DEPAZ

Lima, Perú

2022

**Referencia bibliográfica**

Garcia, M. (2022). *Implementación de una arquitectura micro-frontend para optimizar el desarrollo y despliegue de la aplicación web del sistema de información universitaria de la SUNEDU*. [Trabajo de suficiencia profesional de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática, Escuela Profesional de Ingeniería de Sistemas]. Repositorio institucional Cybertesis UNMSM.

## Metadatos complementarios

| | |
|---|---|
| **Datos de autor** | |
| Nombres y apellidos | MARIO GARCIA CHIHUANHUAYLLA |
| Tipo de documento de identidad | DNI |
| Número de documento de identidad | 71802766 |
| URL de ORCID | https://orcid.org/0000-0003-3143-9684 |
| **Datos de asesor** | |
| Nombres y apellidos | Carlos Edmundo Navarro Depaz |
| Tipo de documento de identidad | DNI |
| Número de documento de identidad | 08482690 |
| URL de ORCID | https://orcid.org/0000-0002-6697-8365 |
| **Datos del jurado** | |
| **Presidente del jurado** | |
| Nombres y apellidos | Cesar Alberto Molina Neyra |
| Tipo de documento | DNI |
| Número de documento de identidad | 40553679 |
| **Miembro del jurado 1** | |
| Nombres y apellidos | Santiago Domingo Moquillaza Henríquez |
| Tipo de documento | DNI |
| Número de documento de identidad | 08280889 |
| **Datos de investigación** | |
| Línea de investigación | No aplica |
| Grupo de investigación | No aplica |
| Agencia de financiamiento | Propio |
| Ubicación geográfica de la investigación | País: Perú<br>Departamento: Lima<br>Provincia: Lima<br>Distrito: Cercado de Lima<br>Jr. Carlos Amezaga No. 375 |

| | |
|---|---|
| | Universidad Nacional Mayor de San Marcos<br>Latitud: -12.0564232<br>Longitud: -77.0843327 |
| Año o rango de años en que se realizó la investigación | 2021 |
| URL de disciplinas OCDE | 2.02.04 -- Ingeniería de sistemas y comunicaciones<br>https://purl.org/pe-repo/ocde/ford#2.02.04 |

***UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS***
***FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA***
***Escuela Profesional de Ingeniería de Sistemas***

***Acta Virtual de Sustentación***
***del Trabajo de Suficiencia Profesional***

*Siendo las 21:01 horas del día 15 de enero del año 2022, se reunieron virtualmente los docentes designados como Miembros de Jurado del Trabajo de Suficiencia Profesional, presidido por el Ing. Molina Neyra Cesar Alberto (Presidente), Mg. Moquillaza Henríquez Santiago Domingo (Miembro) y el Dr. Navarro Depaz Carlos Edmundo (Miembro Asesor), usando la plataforma Meet (https://meet.google.com/jjy-yahj-fza), para la sustentación virtual del Trabajo de Suficiencia Profesional intitulado:* ***"IMPLEMENTACIÓN DE UNA ARQUITECTURA MICRO-FRONTEND PARA OPTIMIZAR EL DESARROLLO Y DESPLIEGUE DE LA APLICACIÓN WEB DEL SISTEMA DE INFORMACIÓN UNIVERSITARIA DE LA SUNEDU",*** *por el Bachiller* ***Garcia Chihuanhuaylla Mario;*** *para obtener el Título Profesional de Ingeniero de Sistemas.*

*Acto seguido de la exposición del Trabajo de Suficiencia Profesional, el Presidente invitó al Bachiller a dar las respuestas a las preguntas establecidas por los miembros del Jurado.*

*El Bachiller en el curso de sus intervenciones demostró pleno dominio del tema, al responder con acierto y fluidez a las observaciones y preguntas formuladas por los señores miembros del Jurado.*

*Finalmente habiéndose efectuado la calificación correspondiente por los miembros del Jurado, el Bachiller obtuvo la nota de* ***19 DIECINUEVE.***

*A continuación el Presidente de Jurados el Ing. Molina Neyra Cesar Alberto, declara al Bachiller* ***Ingeniero de Sistemas****.*

*Siendo las 21:54 horas, se levantó la sesión*
.

***Presidente***
*Ing. Molina Neyra Cesar Alberto*

***Miembro***
*Mg. Moquillaza Henríquez Santiago*

***Miembro Asesor***
*Dr. Navarro Depaz Carlos Edmundo*

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
Universidad del Perú, DECANA DE AMÉRICA
**FACULTAD DE INGENIERIA DE SISTEMAS E INFORMATICA**
**Escuela Profesional de Ingeniería de Sistemas**

# INFORME DE EVALUACIÓN DE ORIGINALIDAD

1. Facultad de Ingeniería de Sistemas e Informática
2. Escuela Profesional de Ingeniería de Sistemas
3. Autoridad académica que emite el informe de originalidad
   **Director de la EPIS**
4. Apellidos y Nombres de la autoridad académica
   **Dr. Luzmila E. Pró Concepción**
5. Operador del programa informático de similitudes
   **Lic. Ulises Roman Concha**
6. Documento evaluado
   **Título de pregrado: "IMPLEMENTACIÓN DE UNA ARQUITECTURA MICRO-FRONTEND PARA OPTIMIZAR EL DESARROLLO Y DESPLIEGUE DE LA APLICACIÓN WEB DEL SISTEMA DE INFORMACIÓN UNIVERSITARIA DE LA SUNEDU"**
7. Autor del documento
   **Bach. Garcia Chihuanhuaylla, Mario**

8. Fecha de recepción del documento 29/11/2021
9. Fecha de aplicación del programa informático de similitudes 06/12/2021
10. Software utilizado
    - Turnitin
11. Configuración del programa detector de similitudes
    - Excluye textos entrecomillados
    - Excluye bibliografía
    - Excluye cadenas menores a 40 palabras
12. Porcentaje de similitudes según programa detector de similitudes 5 **(cinco) %**
13. Fuentes originales de las similitudes encontradas
    Se adjunta en el anexo 1
14. Observaciones
15. Calificación de originalidad
    - Documento cumple criterios de originalidad, sin observaciones [x]
    - Documento cumple criterios de originalidad, con observaciones [ ]
    - Documento no cumple criterios de originalidad [ ]
16. Fecha de informe 06/12/2021

UNMSM
Firmado digitalmente por PRO CONCEPCION Luzmila Elisa FAU 20148092282 soft
Motivo: Soy el autor del documento
Fecha: 20.03.2022 18:29:26 -05:00

Firma de evaluador
***Dra. Luzmila E. Pró Concepción***
*Directora (e) de la EPIS*

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
Universidad del Perú, DECANA DE AMÉRICA
**FACULTAD DE INGENIERIA DE SISTEMAS E INFORMATICA**
**Escuela Profesional de Ingeniería de Sistemas**

## ANEXO 1

# Fuentes originales de las similitudes encontradas

1. cybertesis.unmsm.edu.pe: 3%
2. hdl.handle.net: 1%
2. www.elperulegal.com: 1%
3. Submitted to Universidad del Istmo de Panamá: <1%

UNMSM
Firmado digitalmente por PRO CONCEPCION Luzmila Elisa FAU 20148092282 soft
Motivo: Soy el autor del documento
Fecha: 20.03.2022 18:28:56 -05:00

Firma de evaluador
***Dra. Luzmila E. Pró Concepción***
*Directora (e) de la EPIS*

## DEDICATORIA

Este trabajo va dedicado a mi hijo Matthew, quien es la mayor fuerza y motivación de mi vida y a mi fiel compañera Margarita que apoya todas mis decisiones. También a mis padres y hermanos quienes me enseñaron a ser una persona de bien, en especial a mi padre quien, aunque ya no esté conmigo sé que estaría muy orgulloso de mis logros profesionales.

## AGRADECIMIENTOS

Agradezco a mi asesor, el doctor Carlos Navarro Depaz, quien me compartió sus conocimientos y orientó en el desarrollo de este informe. Le agradezco la paciencia, dedicación y consejos brindados.

**UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS**

**FACULTAD DE INGENIERIA DE SISTEMAS E INFORMATICA**

**ESCUELA PROFESIONAL DE INGENIERIA DE SISTEMAS**

# IMPLEMENTACIÓN DE UNA ARQUITECTURA MICRO-FRONTEND PARA OPTIMIZAR EL DESARROLLO Y DESPLIEGUE DE LA APLICACIÓN WEB DEL SISTEMA DE INFORMACIÓN UNIVERSITARIA DE LA SUNEDU


Autor: Garcia Chihuanhuaylla, Mario

Asesor: Navarro Depaz, Carlos Edmundo



Título: Trabajo de Suficiencia Profesional para optar el Título Profesional de Ingeniero de Sistemas

Fecha: Enero 2022


---

## RESUMEN


En el presente informe de experiencia profesional se describe la implementación de una arquitectura micro-frontend para el sistema de información universitaria de la SUNEDU. El autor se desempeña como arquitecto frontend del equipo de desarrollo del sistema de información universitaria y su participación dentro del proyecto fue desde la concepción de este, identificando las actividades a realizar, así como de liderar e implementar la arquitectura micro-frontend en la aplicación web del sistema. Con la implementación de la arquitectura micro-frontend en el sistema de información universitaria, se logra solucionar el problema de no contar con una arquitectura en el frontend que soporte el crecimiento de la aplicación web del sistema de información universitaria. Esta implementación significó para el equipo de desarrollo la optimización de los tiempos de compilación y facilitar el despliegue de la aplicación web, inconvenientes que ocasionaban retrasos en los cronogramas de desarrollo del sistema.



**Palabras claves:** Arquitectura micro-frontend, frontend.

**NATIONAL MAJOR UNIVERSITY OF SAN MARCOS**

**FACULTY OF SYSTEMS ENGINEERING AND INFORMATIC**

**PROFESSIONAL SCHOOL OF SYSTEMS ENGINEERING**

# IMPLEMENTATION OF A MICRO-FRONTEND ARCHITECTURE TO OPTIMIZE THE DEVELOPMENT AND DEPLOYMENT OF THE WEB APPLICATION UNIVERSITY INFORMATION SYSTEM OF SUNEDU


| | |
|---|---|
| Author: | Garcia Chihuanhuaylla, Mario |
| Advisor: | Navarro Depaz, Carlos Edmundo |
| Title: | Professional Sufficiency Work for opt for the Professional title of Systems Engineer |
| Date: | January 2022 |


---

## ABSTRACT


This professional experience report describes the implementation of a micro-frontend architecture on SUNEDU university information system. The author works as frontend architect of the development team of the university information system and his participation in the project was from its conception, identifying the activities to be carried out, as well as leading and implementing the micro-frontend architecture in the web application of the system. With the implementation of the micro-frontend architecture in the university information system, it is possible to solve the problem of not having an architecture in the front-end that supports the growth of the web application of the university information system. This implementation meant for the development team the optimization of compilation times and facilitating the deployment of the web application, inconveniences that caused delays in the development schedules of the system.



**Keywords:** Micro-frontend architecture, frontend.

# ÍNDICE GENERAL

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# INTRODUCCIÓN

La SUNEDU, creada en el año 2015, es la entidad encargada de velar por el cumplimiento de la ley universitaria y de garantizar la calidad de la educación superior universitaria.

Es en este contexto que la unidad de Documentación e Información Universitaria (UDIU) adscrita a la Dirección de Documentación e Información Universitaria y Registro de Grados y Títulos, es la encargada de liderar el diseño, desarrollo e implementación del sistema de información universitaria SIU, un sistema modular, escalable y adaptativo, el cual será usado por las universidades, instituciones y escuelas de educación superior universitaria para consolidar una fuente de información confiable de la información y procesos universitarios, así como del seguimiento hacia la comunidad docente y universitaria, información que inicialmente era recabada de manera manual.

En el de desarrollo del sistema de información universitaria SIU se trabaja en equipos los cuales se encargan de cada uno de los módulos del sistema.

Conforme avanzaba el proyecto, la aplicación web del SIU llegó a escalar al punto de que los tiempos de compilación eran excesivos, lo cual afectaba negativamente la experiencia durante el de desarrollo, asimismo los despliegues hacia los ambientes de calidad y producción eran complicados ya que la concurrencia de código desarrollado entre los diferentes equipos ocasionaba que la integración de código sea una labor compleja ya que se debía asegurar que todo funcione correctamente luego de la integración, corriendo riesgo de retrasos en el cronograma o incluso la aparición de errores en el ambiente de producción.

Para mitigar esta situación se propuso cambiar la arquitectura de la aplicación frontend en una arquitectura que permita a los diferentes equipos desarrollar de manera independiente y realizar sus despliegues de la misma forma, esto también permitiría mejorar los tiempos de compilación en la etapa de desarrollo al enfocar el desarrollo en pequeñas apps encargadas de cada funcionalidad del sistema.

El presente informa consta de los siguientes capítulos:

En el CAPITULO I, se describe la trayectoria profesional del autor, detallando la experiencia profesional adquirida en las entidades donde laboró, así como las funciones que realizó en estas y los conocimientos técnicos que adquirió a través de los cursos que llevó.

En el CAPITULO II, se detalla la información sobre la SUNEDU, su misión, visión, estructura de la organización resaltando el área en la cual se desarrolló el contexto del proyecto.

En el CAPITULO III, se brinda la información detallada sobre la situación problemática que se presentaba en el desarrollo de la aplicación web del sistema de información universitaria, describiendo la metodología usada y el desarrollo realizado para implementar la arquitectura micro-frontend en la aplicación web.

En el CAPITULO IV, se menciona la reflexión crítica y aporte del autor, realizados una vez implementada la arquitectura micro-frontend en la aplicación web del sistema de información universitaria.

En el CAPITULO V, se hace mención de las conclusiones y de las recomendaciones al informe realizado.

# CAPITULO I
# TRAYECTORIA PROFESIONAL

El autor, bachiller en Ingeniería de Sistemas, cuenta con una experiencia de más de 7 años en el proceso de construcción de soluciones de TI.

Responsable, organizado, autodidacta y con capacidad de trabajo en equipo, es un apasionado de la tecnología y de la búsqueda de soluciones a los problemas con los que se enfrenta en el ámbito laboral; así mismo cuenta con muchas ganas de seguir adquiriendo nuevos conocimientos y compartirlos con su equipo de trabajo.

En las siguientes tablas se detalla la experiencia profesional del autor.

**Tabla 1: Experiencia profesional**

| **Superintendencia nacional de educación superior universitaria (SUNEDU)**<br>**Enero 2019 – Actualidad** | |
|---|---|
| **Área** | Unidad de Documentación e Información Universitaria UDIU |
| **Cargo** | Arquitecto frontend del sistema de información universitaria - SIU |
| **Funciones** | - Implementación de la arquitectura frontend del sistema de información universitaria SIU.<br>- Implementación y mantenimiento de la librería de componentes web utilizados por los sistemas de la SUNEDU.<br>- Apoyo en la implementación de la arquitectura backend del sistema de información universitaria SIU.<br>- Desarrollo de módulos del sistema de información universitaria SIU.<br><br>**Stack backend:** C#, SQL server, Mongo DB, Redis, Rabbit MQ, Docker<br>**Stack frontend:** Angular Framework<br>**Sistema de control de versiones:** GIT y Team Fundation Server (TFS) |

| **Ministerio de la Producción (PRODUCE)**<br>**Agosto 2016 – Diciembre 2018** | |
|---|---|
| **Área** | Oficina de tecnologías de información OTI |
| **Cargo** | Desarrollador de aplicaciones web y móviles |
| **Funciones** | - Desarrollo de las soluciones web y móviles requeridas por la institución, haciendo uso de las buenas prácticas en programación y usando una arquitectura n-capas utilizando .Net Framework.<br>- Garantizar que las soluciones desarrolladas cumplan los requerimientos funcionales y no funcionales.<br>- Mantenimiento de las distintas aplicaciones del Ministerio de la Producción.<br>- Algunos de los proyectos desarrollados fueron los siguientes:<br>• Desarrollo del sistema de trámite documentario<br>• Desarrollo del aplicativo móvil PRODUCE MÓVIL.<br>• Desarrollo del sistema SIFORPA<br>• Desarrollo y mantenimiento del sistema de control sancionador virtual CONSAV.<br><br>**Stack backend:** C#, PHP, SQL server<br>**Stack frontend:** Html, Javascript(jQuery, AngularJS. Angular 2, React, Vue JS), Css, IONIC Framework |
| **Ministerio de la Producción (PRODUCE)**<br>**Noviembre 2013 – Marzo 2016** | |
| **Área** | Oficina de tecnologías de información OTI |
| **Cargo** | Analista Desarrollador de aplicaciones web |
| **Funciones** | - Desarrollo del sistema de control sancionador virtual CONSAV, sistema encargado de apoyar en la gestión de información del procedimiento administrativo sancionador a embarcaciones pesqueras.<br>- Análisis y modelado de base de datos de los nuevos módulos del sistema CONSAV.<br>- Desarrollo y mantenimiento de otros sistemas de información.<br><br>**Stack backend:** PHP y SQL server<br>**Stack frontend:** Html, Javascript(jQuery), Css |

**Nota: Elaboración propia**

**Tabla 2: Formación académica**

| | |
|---|---|
| **Institución** | Universidad nacional mayor de San Marcos |

| | |
|---|---|
| **Periodo** | 2009 - 2014 |
| **Formación recibida** | Grado Académico de Bachiller en Ingeniería de Sistemas Escuela Académico Profesional de Ingeniería de Sistemas– Facultad de Ingeniería de Sistemas e Informática – Universidad Nacional Mayor de San Marcos. |

**Nota: Elaboración propia**

**Tabla 3:Cursos**

| | |
|---|---|
| **Microservicios Arquitectura - Desarrollo** | |
| **Año** | Febrero 2020 |
| **Institución** | B. SOFT Group |
| **Arquitecto desarrollador plataforma web Net Core** | |
| **Año** | Junio 2019 |
| **Institución** | B. SOFT Group |
| **Desarrollo y despliegue de soluciones en capas con C#** | |
| **Año** | Enero 2017 |
| **Institución** | Sistemas UNI |

**Nota: Elaboración propia**

# CAPÍTULO II

# CONTEXTO EN EL QUE SE DESARROLLÓ LA EXPERIENCIA

## 2.1. EMPRESA – ACTIVIDAD QUE REALIZA

La SUNEDU es el ente que licencia, supervisa y fiscaliza el servicio educativo superior universitario.

De acuerdo con el portal del gobierno peruano (2021) nos manifiesta lo siguiente:

> Sunedu asegura una oferta educativa de calidad en favor de los estudiantes, a través del licenciamiento y supervisión de este servicio público, con eficiencia, predictibilidad, transparencia y respeto a la autonomía universitaria.
>
> A través de la publicación de la Ley Universitaria, Ley Nº 30220, se hace oficial la creación de la Superintendencia Nacional de Educación Superior Universitaria (Sunedu), cuya constitucionalidad fue ratificada por el Tribunal Constitucional el 26 de enero de 2016.
>
> Este organismo público nace para proteger el derecho de los jóvenes a recibir una educación universitaria de calidad y, de esta manera, mejorar sus competencias profesionales.
>
> La SUNEDU se convirtió –desde el 5 de enero de 2015– en la responsable del licenciamiento para ofrecer el servicio educativo superior universitario. Siendo un organismo público técnico especializado, adscrito al Ministerio de Educación, se encarga también de verificar el cumplimiento de la Condiciones Básicas de Calidad y fiscalizar si los recursos públicos y los beneficios otorgados a través del marco legal son destinados hacia fines educativos y el mejoramiento de la calidad.

La SUNEDU asume la función de administrar el Registro Nacional de Grados y Títulos, bajo la consigna de brindar seguridad jurídica de la información que se encuentra registrada y garantizar su autenticidad.

## 2.2. MISIÓN

"Asegurar una oferta educativa de calidad en favor de los estudiantes, a través del licenciamiento y supervisión de este servicio público, con eficiencia. predictibilidad. transparencia y respeto a la autonomía universitaria." (Sunedu, 2021)

## 2.3. VISIÓN

De acuerdo con la web de la SUNEDU (Sunedu, 2021) nos manifiesta:

Garantizar educación de calidad para los peruanos, que les permite desarrollar su potencial y convertirse en ciudadanos que valoran su cultura, conocen sus derechos y responsabilidades, desarrollan sus talentos y participan de manera innovadora, competitiva y comprometida en las dinámicas sociales, contribuyendo al desarrollo de sus comunidades y del país en su conjunto.

## 2.4. ORGANIZACIÓN DE LA EMPRESA

A continuación, se muestra el organigrama de la SUNEDU, actualizado a octubre 2021, resaltando el área donde se llevó a cabo la experiencia profesional.

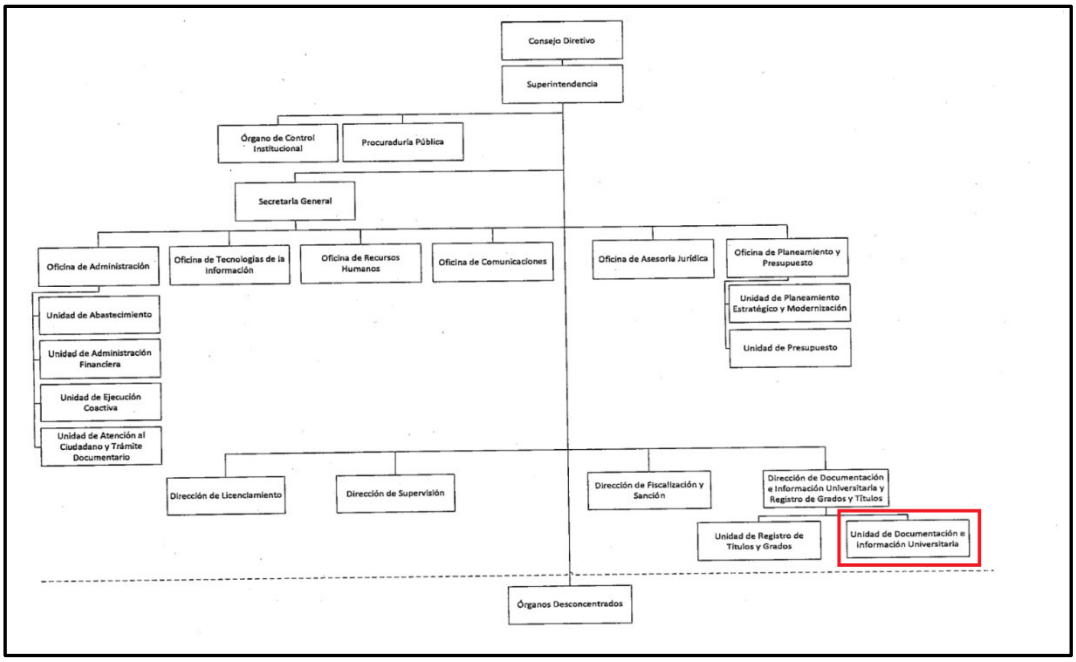



**Figura 1: Organigrama de la SUNEDU**

**Nota: Adaptado de web de la SUNEDU** (2021)

## 2.5. ÁREA, CARGO Y FUNCIONES DESEMPEÑADAS

El autor de este informe se desempeña como arquitecto frontend del sistema de información universitaria SIU en la Unidad de Documentación e Información Universitaria.

Las funciones que presenta el autor están alineadas a lo siguiente:

- Implementar y mejorar la arquitectura frontend del sistema de información universitaria SIU.
- Elaborar los componentes web necesarios para las funcionalidades requeridas en el sistema de información universitaria SIU
- Apoyar en la implementación de la arquitectura backend del sistema de información universitaria SIU.
- Desarrollo de funcionalidades para el sistema de información universitaria SIU.

## 2.6. EXPERIENCIA PROFESIONAL REALIZADA EN LA ORGANIZACIÓN

El autor como arquitecto frontend y asistente de arquitecto Backend tuvo las siguientes experiencias.

- Implementó la arquitectura frontend del sistema de información universitaria SIU.
- Implementó, publicó y realizó el mantenimiento de la librería de componentes web utilizados por los sistemas de la SUNEDU.
- Apoyó en la implementación de la arquitectura backend del sistema de información universitaria SIU.
- Propuso una arquitectura frontend basada en microservicios para optimizar los tiempos de desarrollo y despliegue.
- Realizó la migración de versión del framework angular del sistema de información universitaria SIU, desde la versión 7 a la versión 12.

# CAPÍTULO III
# ACTIVIDADES DESARROLLADAS
## 3.1. SITUACIÓN PROBLEMÁTICA

Debido a que al iniciar el desarrollo del sistema de información universitaria se optó por utilizar una arquitectura de microservicios, se pensó que esta arquitectura elegida permitiría al proyecto escalar adecuadamente, mitigando riesgos de acoplamiento de código y otros problemas relacionados con una arquitectura monolítica. Sin embargo, mientras en el backend con los microservicios "dividimos" nuestro desarrollo en pequeñas piezas de negocio que serán orquestadas según las necesidades de la aplicación, en el frontend poco a poco se iba construyendo un gran monolito, lo que desde un principio se pretendió evitar.

El equipo de desarrollo del sistema de información universitaria SIU, se encuentra conformado por sub equipos encargados de los diferentes módulos, funcionalidades o mantenimientos del sistema. La aplicación web frontend del SIU se encuentra desarrollada bajo el framework Angular versión 7 y las versiones del código se administran bajo el controlador de versiones. Durante el proceso de desarrollo bajo el framework Angular es necesario que se compilen todos los archivos cada vez que se hace un cambio en el código fuente, ya que es la manera en que el framework trabaja, a medida de que el proyecto iba creciendo la aplicación web tenía más archivos que compilarse, esto repercutía en el tiempo que tomaba el desarrollo ya que un simple cambio en el desarrollo podía tomar hasta 2 o 3 minutos (se llegó ver que en ordenadores con procesadores más antiguos podía llegar a los 5 minutos) en mostrarse en pantalla, esto dependiendo del hardware del ordenador en la que se está trabajando. Situación que producía riesgos en los cronogramas de desarrollo debido al excesivo tiempo perdido en esperar visualizar los cambios que uno desarrolla. Por otro lado, los despliegues de la aplicación web resultaban ser tareas muy tediosas, llegando a tomar una sola compilación en modo release de 15 a 25 minutos. Otra situación existente era la integración que se tenía que realizar antes de cada despliegue. Al volverse la aplicación web frontend un gran monolito, cada vez que se desplegaba un módulo, el código de este se tenía que

integrar con el que ya estaba en funcionamiento, teniéndose que validar el correcto funcionamiento de toda la aplicación para verificar que luego de la integración de código no se afecten las funcionalidades existentes. Esta situación repercutía en los tiempos en el cronograma de despliegue o incluso suponía el riesgo de encontrar errores de aplicación en el ambiente de producción.

### 3.1.1. Definición del problema

#### 3.1.1.1. Problema General

El principal problema radica en la carencia de una arquitectura en la aplicación web frontend del sistema de información universitaria de la SUNEDU que soporte el crecimiento de esta, sin comprometer los tiempos de compilación de la aplicación ni perjudicar a las funcionalidades existentes a la hora de hacer el despliegue de algún módulo.

Al tener una arquitectura monolítica en la capa web frontend, surgen inconvenientes durante la etapa de desarrollo de los módulos y los mantenimientos debido a que se vuelve complicada la administración de todo el código fuente de la aplicación ya que todos los proyectos están interconectados haciendo que los equipos sean dependientes entre sí pese a que las funcionalidades que desarrollan no lo son. Asimismo, al ser un proyecto con muchos archivos, el tiempo de compilación en modo desarrollo o modo producción se ve afectado directamente. Este tiempo elevado se debe a que al realizar cualquier cambio en la aplicación al momento de desarrollar el compilador tendrá que analizar todos los archivos que conforman la aplicación web.

El tiempo que se toma sólo en esperar que la aplicación compile ya sea al momento de desarrollar o al momento de generar el modo release para realizar los despliegues a ambientes productivos son excesivos y afectan al cronograma del proyecto.

¿De qué manera se puede optimizar el desarrollo y despliegue de la aplicación web frontend del sistema de información universitaria de la SUNEDU?

#### 3.1.1.2. Problemas Específicos

- ¿Cómo elegir la mejor tecnología para implementar la arquitectura micro-frontend en la aplicación web frontend del sistema de información universitaria de la SUNEDU?
- ¿Cómo disminuir los tiempos de compilación en la fase de desarrollo de la aplicación web frontend del sistema de información universitaria de la SUNEDU?
- ¿Cómo optimizar el procedimiento de despliegue de la aplicación web frontend del sistema de información universitaria de la SUNEDU?

## 3.2. SOLUCIÓN

Para solucionar el problema planteado se propone la implementación de una arquitectura MICRO-FRONTEND en la aplicación web del sistema de información universitaria de la SUNEDU.

### 3.2.1. Objetivos

#### 3.2.1.1. Objetivo General

Implementar una arquitectura MICRO-FRONTEND para optimizar el desarrollo y despliegue de la aplicación web del sistema de información universitaria de la SUNEDU.

#### 3.2.1.2. Objetivos Específicos

- Identificar la mejor tecnología a utilizar para implementar la arquitectura micro-frontend en la aplicación web frontend del sistema de información universitaria de la SUNEDU.
- Dividir la aplicación web frontend del sistema de información universitaria de la SUNEDU en pequeñas apps que sean más ligeras para su compilación.
- Crear sitios web en el servidor para poder desplegar independientemente cada una de las aplicaciones micro-frontend del sistema de información universitaria de la SUNEDU.

### 3.2.2. Alcance

#### 3.2.2.1. Alcance funcional

El alcance del proyecto contempla realizar la migración de arquitectura monolítica de la capa web a una arquitectura micro-frontend en los módulos del sistema de información universitaria, los cuales se listan a continuación:

- Módulo de Autoridades
- Módulo de emisión de carnés
- Módulo de grados académicos
- Módulo de Renati
- Módulo de reportes
- Módulo de seguimiento académico
- Módulo de configuración
- Módulo de mallas curriculares
- Módulo de seguimiento de docentes
- Módulo institucional
- Módulo de gestión de personal
- Módulo de soporte
- Módulo de supervisión

#### 3.2.2.2. Alcance Organizacional

El proyecto permitió que el equipo de desarrollo del SIU, perteneciente a la Unidad de Documentación e Información Universitaria UDIU, optimizara el procedimiento de desarrollo y despliegue de la aplicación web. El equipo de desarrollo del SIU se encuentra conformado de la siguiente forma:

- **Jefe de la UDIU:** Es el jefe de la Unidad de Documentación e Información Universitaria.
- **Coordinadora del SIU:** Es la encargada de realizar las gestiones necesarias para que el proyecto se desarrolle con normalidad y quien representa a todo el equipo de encargado del desarrollo del Sistema de Información Universitaria.
- **Arquitecto del SIU:** Es el profesional encargado de la construcción de la arquitectura de toda la aplicación y orquestar la comunicación de las aplicaciones backend y frontend, también de velar por la disponibilidad de la aplicación bajo toda demanda.
- **Analistas del SIU:** Es el equipo de profesionales encargado de levantar los requerimientos del sistema.
- **Arquitecto Frontend del SIU:** Es el profesional encargado de armar la arquitectura de la aplicación frontend del SIU y de brindar soporte al equipo de

desarrollo frente a los inconvenientes que surjan durante desarrolla de la aplicación web.

- **Equipo de mantenimiento del SIU:** Es el equipo de profesionales que se encargan de desarrollar las mejoras a los módulos existentes del SIU.
- **Equipo de data y BI del SIU:** Es el equipo de profesionales que administran las bases de datos del SIU y se encargan de realizar el proceso de BI a estas.

En la siguiente figura se presenta la estructura del equipo de desarrollo del Sistema de Información Universitaria.

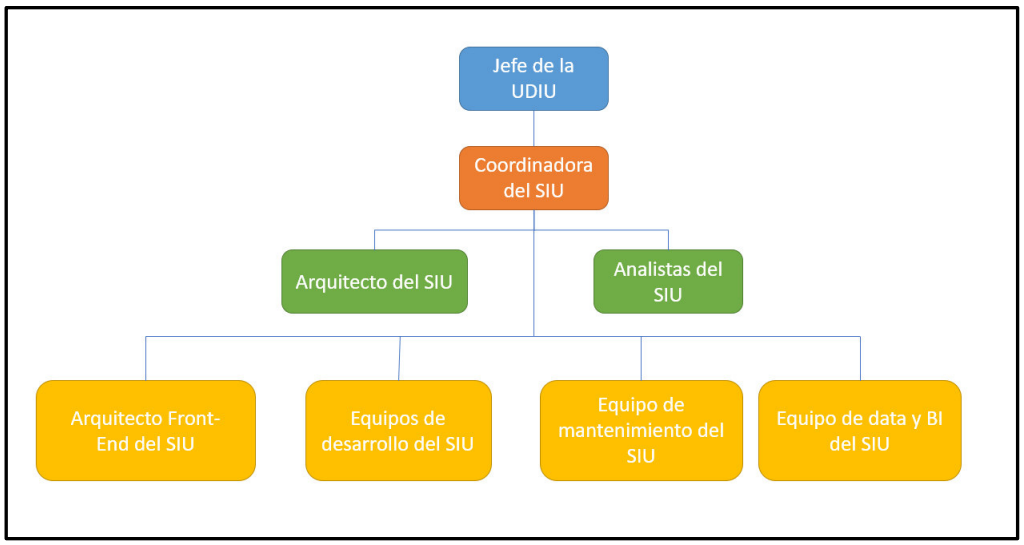



**Figura 2: Organización del equipo de desarrollo del SIU**

**Nota: Elaboración propia**

#### 3.2.2.3. Alcance Geográfico

El proyecto de implementación de la arquitectura micro-frontend, se realiza sobre el Sistema de Información Universitaria SIU, el cual abarca la información de todas las universidades (nacionales y particulares) e instituciones de educación superior universitaria a nivel nacional.

### 3.2.3. Etapas y metodología

El proyecto de implementación de la arquitectura micro-frontend en el Sistema de Información Universitaria fue realizado bajo la metodología SCRUM y se tuvieron las siguientes consideraciones:

- El proyecto tuvo una duración de casi 11 semanas, desde el 16 de agosto hasta el 29 de octubre.

- Los Sprints tuvieron una duración de 1 a 2 semanas.
- El autor del presente informe estuvo presente desde el inicio hasta el final de proyecto como parte del Scrum Team bajo el rol de arquitecto frontend.

En la siguiente tabla se presenta el cronograma de las iteraciones realizadas.

**Tabla 4: Cronograma de actividades por sprint**

| Sprint | Actividad | Nro. Días Hábiles | Inicio | Fin | Responsables |
|---|---|---|---|---|---|
| 1 | Sprint Planning | 1 | 16/08/2021 | 16/08/2021 | Product Owner<br>Scrum Master<br>Scrum Team |
| | Desarrollo del Sprint | 9 | 17/08/2021 | 27/08/2021 | Scrum Team |
| 2 | Sprint Planning | 1 | 31/08/2021 | 31/08/2021 | Product Owner<br>Scrum Master<br>Scrum Team |
| | Desarrollo del Sprint | 3 | 01/09/2021 | 03/09/2021 | Scrum Team |
| 3 | Sprint Planning | 1 | 06/09/2021 | 06/09/2021 | Product Owner<br>Scrum Master<br>Scrum Team |
| | Desarrollo del Sprint | 4 | 07/09/2021 | 10/09/2021 | Scrum Team |
| 4 | Sprint Planning | 1 | 13/09/2021 | 13/09/2021 | Product Owner<br>Scrum Master<br>Scrum Team |
| | Desarrollo del Sprint | 14 | 14/09/2021 | 01/10/2021 | Scrum Team |
| 5 | Sprint Planning | 1 | 04/10/2021 | 04/10/2021 | Product Owner<br>Scrum Master<br>Scrum Team |
| | Desarrollo del Sprint | 8 | 05/10/2021 | 15/10/2021 | Scrum Team |
| 6 | Sprint Planning | 1 | 18/10/2021 | 18/10/2021 | Product Owner<br>Scrum Master |

| | | | | |
|---|---|---|---|---|
| | | | | Scrum Team |
| Desarrollo del Sprint | 9 | 19/10/2021 | 29/10/2021 | Product Owner<br>Scrum Team |

**Nota: Elaboración propia**

### 3.2.4. Fundamentos utilizados

#### 3.2.4.1. Metodología Scrum

Según Deemer, Benefield, Larman, y Vodde (2009), Scrum es un marco de trabajo ágil iterativo e incremental que sirve para el desarrollo de proyectos. La forma de desarrollo es a través de ciclos llamados Sprints. Cada Sprint tiene una duración de 1 a 4 semanas y son consecutivos uno de otro.

**Roles de Scrum**

- **Product Owner:** Es el encargado de elaborar la lista de funcionalidades del producto, identificándolas y priorizando cuáles deberán ir al principio de la lista, refinando este listado de manera continua.
- **Scrum Master:** Es el encargado de guiar al equipo hacia el éxito y de asegurarse que este aprende y aplique Scrum correctamente
- **Scrum Team:** Es el equipo encargado de construir el producto.

**Fases de Scrum**

- **Sprint:** Es la etapa iterativa en la que se construirá el producto definido para ese sprint cuya duración máxima es de un mes.
- **Sprint Planning:** Es la reunión en la que se define el objetivo y las tareas a realizar dentro de los sprint. Se define lo que se va a hacer en cada sprint según el Product Backlog y las tareas que se realizarán para completar cada ítem seleccionado del Product Backlog.
- **Daily meeting:** Es la reunión diaria de un máximo de 15 minutos que servirá para inspeccionar el trabajo. El equipo de desarrollo debe contestar a las preguntas ¿Qué hice ayer?, ¿Qué haré hoy? y ¿Qué impedimentos necesito que se solucionen?
- **Sprint Review:** Es la reunión que se realiza al final de cada sprint en donde se presentará lo desarrollado al cliente. El cliente valida el funcionamiento del

desarrollo y ofrece el feedback sobre nuevas tareas o funcionalidades que se tendrán que agregar al Product Backlog.

- **Sprint Retrospective:** Es la reunión del equipo en la que se evalúa cómo se implementó Scrum durante el último Sprint y proponiendo así mejoras para el siguiente.

**Artefactos Scrum**

- **Product Backlog:** "Es el inventario en el que se almacenan todas las funcionalidades o requisitos en forma de lista priorizada. Estos requisitos serán los que tendrá el producto o los que irá adquiriendo en sucesivas iteraciones" (Trigas Gallego, 2012).
- **Sprint Backlog:** "Es la lista de tareas que elabora el equipo durante la planificación de un sprint. Se asignan las tareas a cada persona y el tiempo que queda para terminarlas" (Trigas Gallego, 2012)**.**

#### 3.2.4.2. HTML

Según Eslava Muñoz (2012), HTML (HyperText Markup Language) se define como un lenguaje de marcado para la creación de páginas web que se utiliza para definir y traducir la estructura e información en forma de texto, así como para complementar el texto con imágenes o tablas.

#### 3.2.4.3. Iframe

Iframe, abreviatura de Inline Frame, es un elemento de HTML que permite insertar dentro de este, documentos, videos y otros medios interactivos para poder ser visualizados en cualquier lugar dentro de una página web.

#### 3.2.4.4. Javascript

Javascript es un lenguaje de programación interpretado, ligero y dinámico que es utilizado mayormente en navegadores web, pero también en otros entornos como servidores web o aplicaciones móviles.

> "Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar

directamente en cualquier navegador sin necesidad de procesos intermedios." (Pérez, 2009)

#### 3.2.4.5. Typescript

Según su página web oficial, "Typescript es un lenguaje de programación fuertemente tipado basado en javascript, y que ofrece mejores herramientas a cualquier escala." (Typescript, 2021)

Desarrollado por Microsoft publicado en 2012. Este lenguaje opensource es un superconjunto de javascript que agrega mayores funcionalidades como el tipado estático y objetos basados en clases, así como otros conceptos de la programación orientada a objetos. Una vez realizada la compilación del código typescript el resultado obtenido será código javascript que podrá ser ejecutado en cualquier navegador o servidor que corra javascript.

#### 3.2.4.6. Protocolo HTTP

"HTTP es un protocolo que permite compartir información mediante el modelo cliente-servidor y sin estado, esto es, el cliente realiza una solicitud al servidor y espera una respuesta de este." (Luna Del Águila, 1999)

#### 3.2.4.7. REST

Según Doglio (2015), REST es un estilo de arquitectura definido para crear y organizar sistemas distribuidos. No es una guía ni un estándar o algo que implique que haya reglas a seguir para construir una arquitectura RESTful. La principal idea de REST es que un sistema distribuido que aplique REST mejorará siguientes aspectos:

**Rendimiento**: El estilo de comunicación propuesta por REST, supone ser simple y eficiente permitiendo incrementar el rendimiento de las aplicaciones que lo adopten.

**Escalabilidad**: La simple interacción propuesta por REST contribuye a mejorar este aspecto.

**Interface simple**: Ofrece una interface que simplifica la interacción entre sistemas.

**Modificabilidad**: La separación de conceptos propuestos por REST hace posible que los componentes del sistema sean modificados con el mínimo costo y riesgo.

**Portabilidad**: REST es agnóstico del lenguaje utilizado, lo cual permite ser implementado y consumido por cualquier tipo de tecnología.

**Recuperabilidad:** El concepto Stateless que propone REST, permite la fácil recuperación del sistema ante una falla.

Las condiciones que debe cumplir una arquitectura REST son las siguientes:

**Cliente – Servidor: "**Un servidor es el encargado de administrar un conjunto de servicios y las solicitudes relacionadas a estos servicios. Estas solicitudes se realizan a través de un sistema cliente que necesita de estos servicios". (Doglio, 2015)

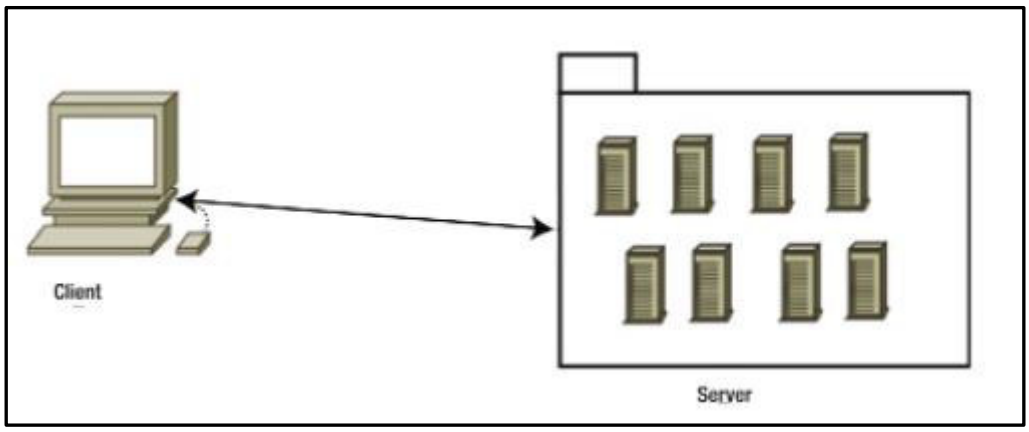



**Figura 3: Diagrama de arquitectura cliente servidor**

**Nota: Adaptado de Doglio** (2015)

**Stateless:** Stateless, que en español quiere decir "sin estado", significa que durante la comunicación entre cliente y servidor, cada solicitud realizada por el cliente debe contener toda la información que necesita el servidor para poder ser procesada, sin que tenga que usar ningún dato almacenado, dice Doglio (2015).

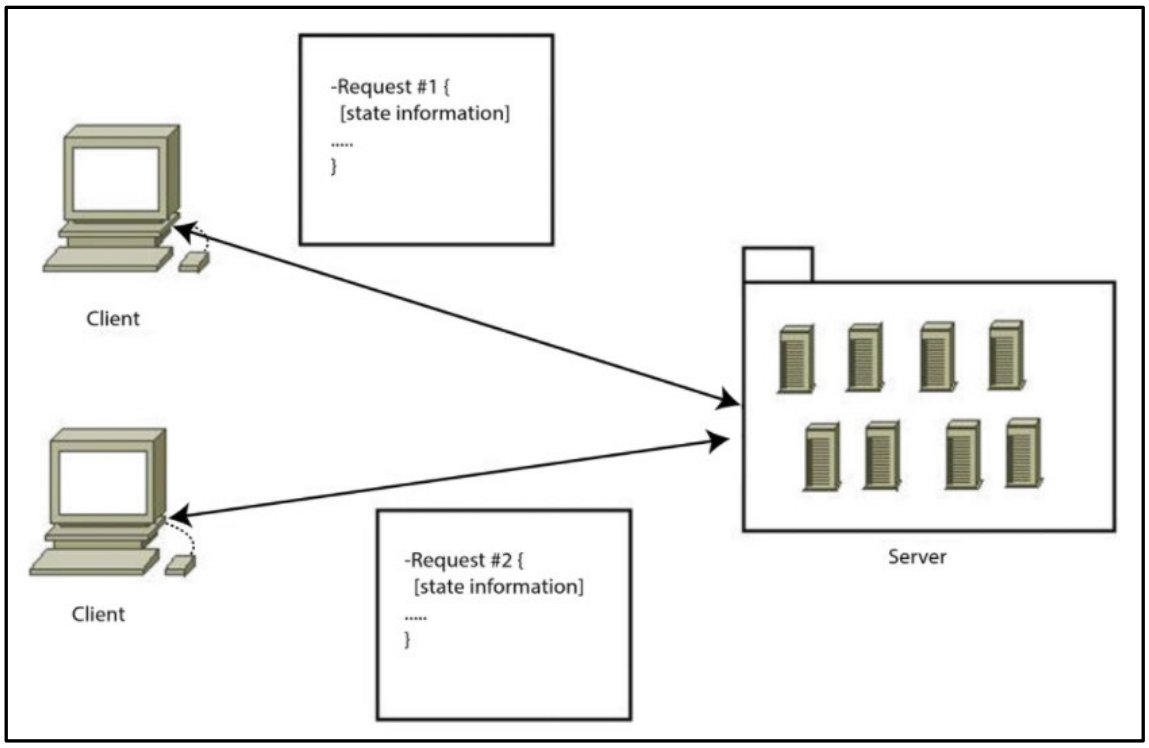



**Figura 4: Representación de la condición Stateless**

**Nota: Adaptado de Doglio** (2015)

**Cacheable:** Cada respuesta a una solicitud debe poder ser almacenada en caché. Esto para ofrecer una mejor percepción de rendimiento al cliente.

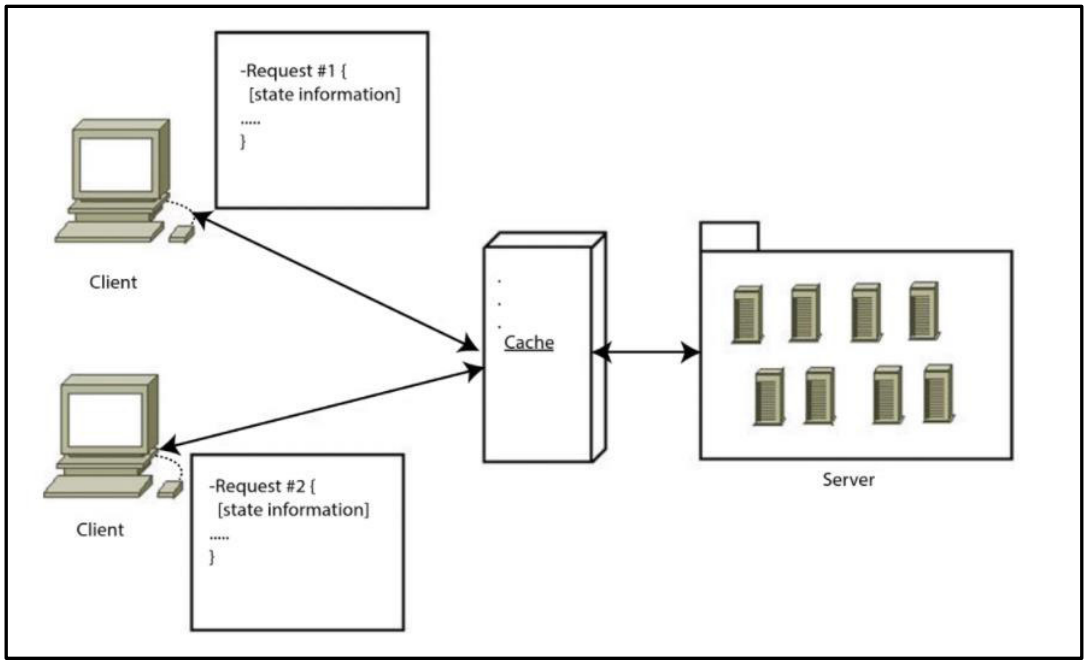



**Figura 5: Representación de una interacción entre cliente y caché de un servidor stateless**

**Nota: Adaptado de Doglio** (2015)

**Interface Uniforme:** Aplicando este concepto se simplifica el trabajo al cliente al momento de interactuar con el sistema.

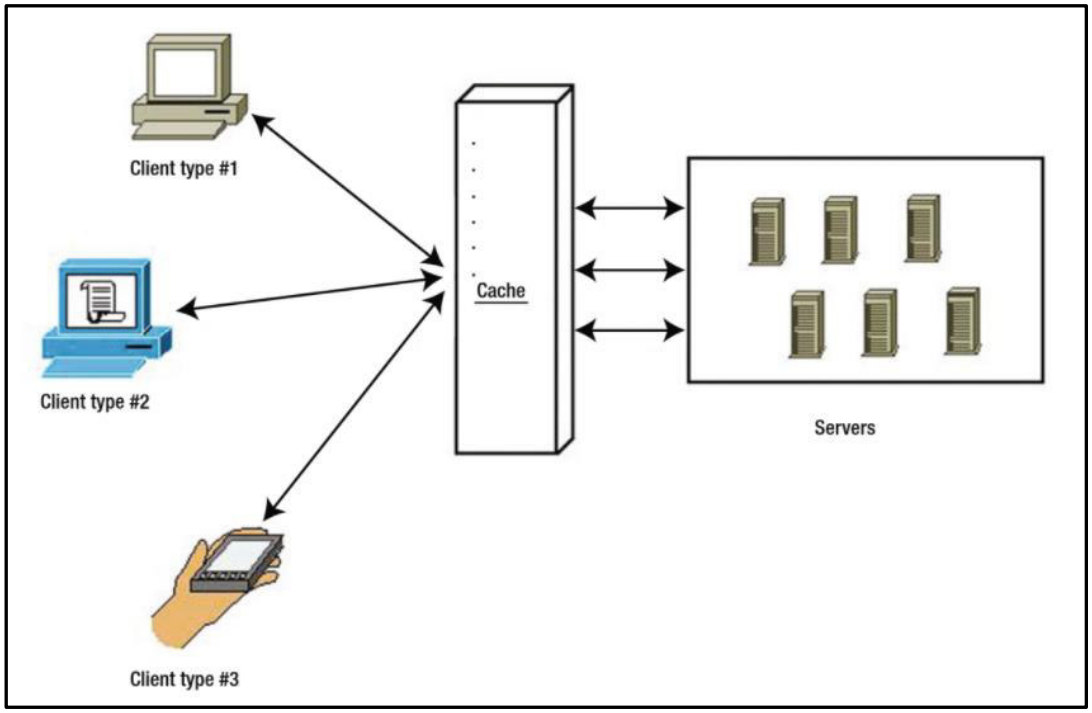



**Figura 6: Diferentes clientes interactuando de la misma manera con el servidor gracias a una interface uniforme**

**Nota: Adaptado de Doglio** (2015)

**Arquitectura en capas:** Para soportar la alta demanda de solicitudes, REST propone una arquitectura en capas en la cual cada capa superior se comunica con la siguiente capa inferior, de esta manera también se simplifica la complejidad del sistema y se reduce el acoplamiento entre componentes del propio sistema.

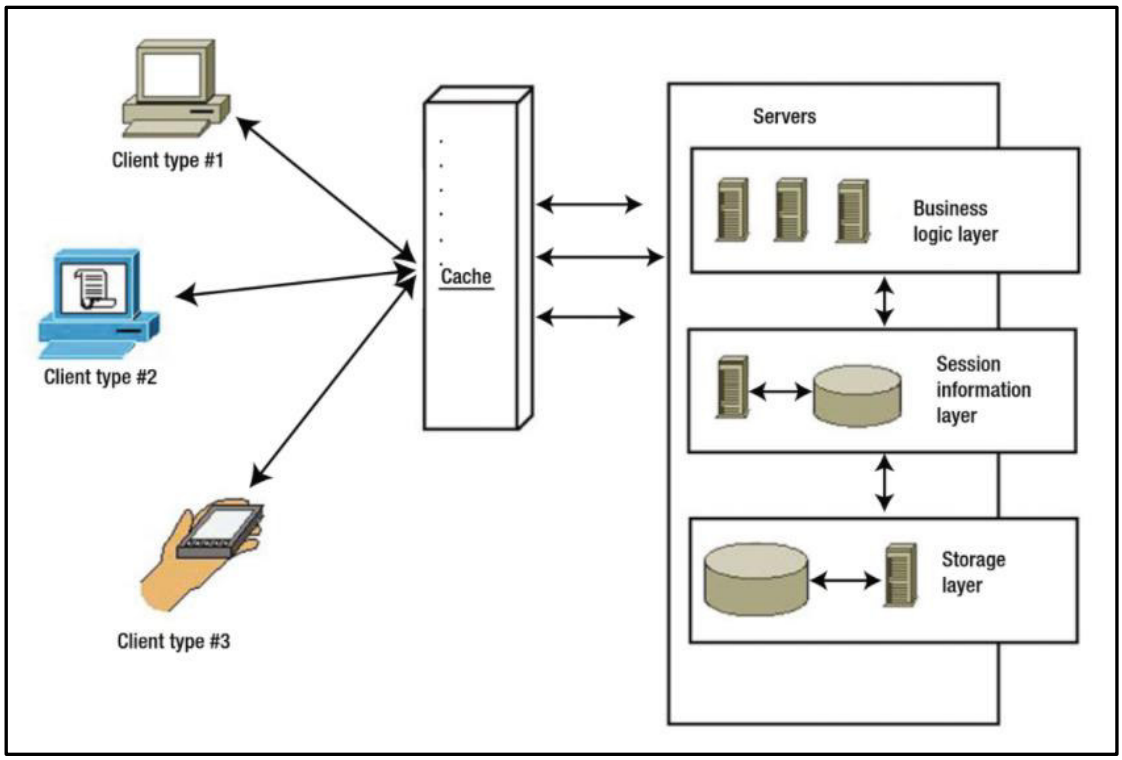



**Figura 7: Ejemplo de arquitectura multicapa**

**Nota: Adaptado de Doglio** (2015)

REST utiliza 4 tipos de operaciones para el acceso a los recursos:

- Read (Lectura): Esta operación permite obtener información de un recurso.
- Create (Escritura): Esta operación permite crear un nuevo recurso en el sistema.
- Delete (Eliminar): Esta operación permite eliminar un recurso del sistema.
- Update (Actualizar): Esta operación permite modificar un recurso del sistema.

Las solicitudes para acceso a los recursos se envían a través del protocolo HTTP utilizando los siguientes verbos:

- GET: Operaciones de lectura.
- POST: Operaciones de creación.
- PUT: Operaciones de modificación.
- DELETE: operaciones de eliminación.

### 3.2.4.8. API REST

Un API (Application Programming Interface) es la interfaz de programación de aplicaciones y permite la comunicación entre sistemas sin importar la tecnología sobre la que estén construidas. Una Api REST por lo tanto permite la comunicación entre sistemas usando el estilo propuesto por REST.

### 3.2.4.9. GIT

> "Git es un sistema de control de versiones de código, gratuito y open-source diseñado para manejar desde proyectos pequeños hasta proyectos muy grandes" (Git, 2021).

Git es una herramienta que administra las versiones del código de forma distribuida y permite a los desarrolladores tener un registro de todos los cambios realizados en cada archivo de un proyecto.

### 3.2.4.10. JSON

Siglas de Javascript Object Notation, es un formato muy popular utilizado para la transferencia de datos, es de fácil entendimiento para el ser humano y simple de ser generado por los sistemas de información. Es agnóstico del lenguaje y por esta razón resulta ideal para el intercambio de datos en apis rest.

> "JSON es diseñado para ser un lenguaje de intercambio de datos el cual es legible para el ser humano y fácil de interpretar y usar para los computadores". (Nurseitov, Paulson, Reynolds, & Izurieta, 2009)

En la siguiente figura se aprecia un ejemplo del formato JSON.

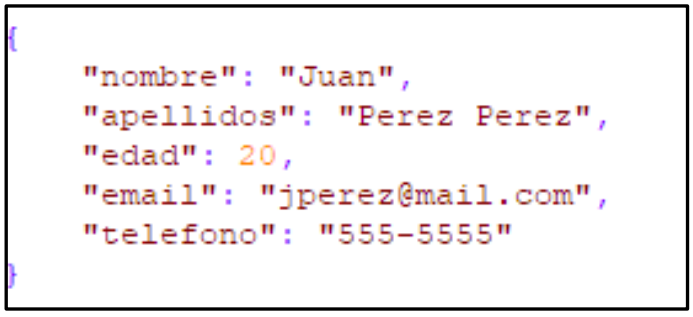



**Figura 8: Ejemplo del formato JSON**

**Nota: Elaboración propia**

#### 3.2.4.11. Node JS

> "Node.js® es un entorno de ejecución para JavaScript construido con V8, motor de JavaScript de Chrome." (Node.js, 2021)

Node JS permite ejecutar aplicaciones en escritas en javascript desde un servidor.

#### 3.2.4.12. NPM

Según su página oficial, "NPM es el registro de software más grande del mundo en donde desarrolladores de cada continente comparten paquetes y muchas organizaciones la usan para administrar sus desarrollos privados". (NPM, 2021)

> "NPM permite registrar nuestros paquetes con un nombre que podemos importar y exportar a través de este nombre. El registro de npm es un web service que aloja los paquetes y las hace accesibles en toda la internet." (Ali, 2013)

#### 3.2.4.13. Framework Angular

Angular es un framework opensource para la construcción y desarrollo de aplicaciones SPA desarrollado por Google en el año 2016.

Según su página web (2021), angular se define como una plataforma de desarrollo construida Typescript, la cual incluye un framework basado en componentes que hace posible construir aplicaciones web escalables, un conjunto de librerías que cubren una amplia variedad de funcionalidades (manejo de rutas, administración de formularios, comunicación cliente servidor y más) y un gran conjunto de herramientas que ayudarán en el desarrollo, despliegue y el testing de la aplicación.

Angular es un framework modular, que permite la escalabilidad de las aplicaciones web y mantiene el código desarrollado ordenadamente gracias a su patrón MVC (Modelo-Vista-Controlador). Actualmente se encuentra en su versión 13 y es uno de los frameworks de desarrollo web más populares en la actualidad. Es altamente recomendable para proyectos de todo tipo de dimensión ya que cuenta con todo lo necesario para la construcción de aplicaciones robustas, escalables y de calidad.

**Conceptos Clave del Framework**

- **Módulos:** Un Módulo en Angular declara un contexto para un grupo de componentes que se dedican a una funcionalidad específica. Agrupa componentes, servicios, directivas, etc.
- **Componentes:** Cada componente de angular es una clase que define la lógica de aplicación y datos de la funcionalidad que controla y se asocia con un archivo html.
- **Templates:** Un template de Angular combina HTML y sintaxis de Angular para modificar los elementos HTML antes de ser visualizados.
- **Directivas:** Una directiva de Angular ofrece la capacidad de integrar lógica programada a los elementos HTML, así como manipular estos elementos HTML.
- **Data Binding:** Existen 2 tipos de data binding en Angular. El primero, Event Binding, permite responder a la aplicación ante la acción de un usuario sobre un componente actualizando los datos de la aplicación. El segundo, Property Binding, permite introducir valores que se van calculando en la aplicación hacia el elemnto HTML.
- **Servicios:** Sirven para compartir lógica que no está asociada a una vista.
- **Inyección de Dependencias:** Permiten a los componentes de Angular mantenerse limpias y eficientes inyectando las dependencias que esta requiera.
- **Routing:** Angular proporciona un servicio que define una ruta de navegación entre los diferentes estados de la aplicación, basado en las convenciones de navegación comúnmente usadas en el navegador.

En la siguiente figura se observa la arquitectura de Angular, utilizando los conceptos descritos anteriormente.

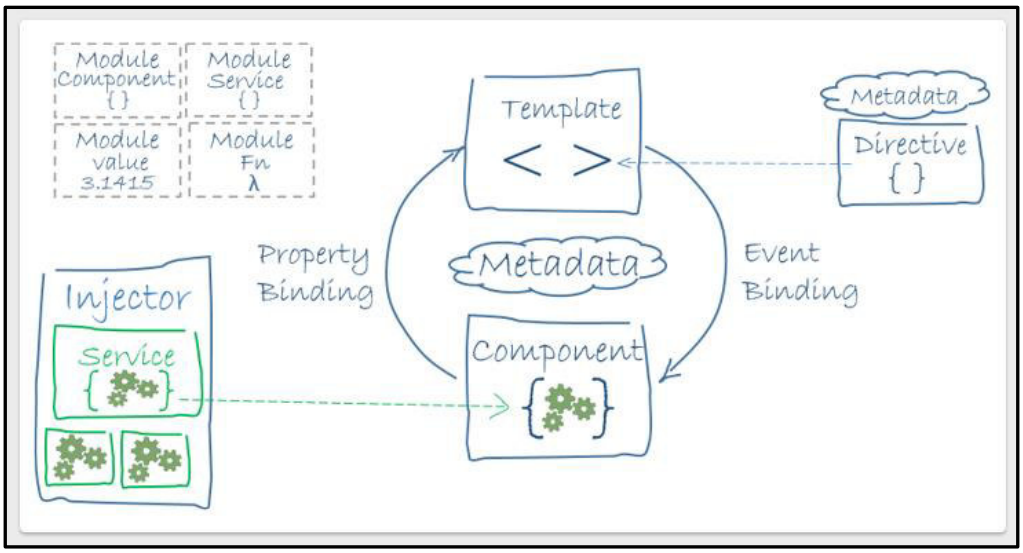



**Figura 9: Arquitectura de Angular**

**Nota: Adaptado de la página web oficial de Angular** (2021)

### 3.2.4.14. Webpack

Según su web oficial, "Webpack es un empaquetador de módulos cuyo principal propósito es empaquetar archivos JavaScript para ser utilizados en el navegador web, sin embargo, es capaz de transformar, agrupar o empaquetar casi cualquier otro tipo de recurso." (Webpack, 2021)

Webpack nos permite generar un único archivo con todos los módulos JavaScript que requiere nuestra aplicación para funcionar, además, realiza las tareas de minificación y compresión del código, las cuales optimizan el código desarrollado.

**Conceptos Clave de Webpack**

- **Entry Point:** Son los archivos que webpack procesara para generar los paquetes o archivos resultantes (*.bundle.js).
- **Output:** Es la ruta donde se alojará el paquete o módulo generado.
- **Loaders:** Webpack solo entiende archivos JavaScript y JSON, los loaders son rutinas que permiten a webpack entender y procesar otros tipos de archivos y convertirlos en módulos válidos que puedan ser consumidos por la aplicación.
- **Plugins:** Los plugins se utilizan para realizar tareas de optimización en los paquetes, administrar los archivos de recursos o inyectar variables de entorno.

- **Webpack Module Federation:** Es una tecnología integrada en la versión 5 de webpack, la cual brinda la opción de integrar módulos webpack de una aplicación frontend en tiempo de ejecución y de forma remota.

### 3.2.4.15. Aplicación SPA

Según Jadhav, Sawant y Deshmukh (2015), Una aplicación SPA (single page application), es una aplicación web que carga en el navegador todos sus recursos durante la carga inicial, y cuyas páginas son reemplazadas por otras dependiendo de la interacción con el usuario. Un SPA está compuesta de componentes individuales que pueden ser reemplazados independientemente sin necesidad de recargar toda la página.

### 3.2.4.16. Arquitectura de Microservicios

Según López y Maya (2017), la arquitectura de microservicios es un enfoque para desarrollar aplicaciones como un conjunto de pequeños servicios que se ejecutan dentro de sus propios procesos y mecanismos ligeros de comunicación, a menudo estos servicios vienen a ser Apis sobre el protocolo HTTP. Estos servicios se construyen alrededor de las capacidades del negocio y con independencia para su despliegue e implementación.

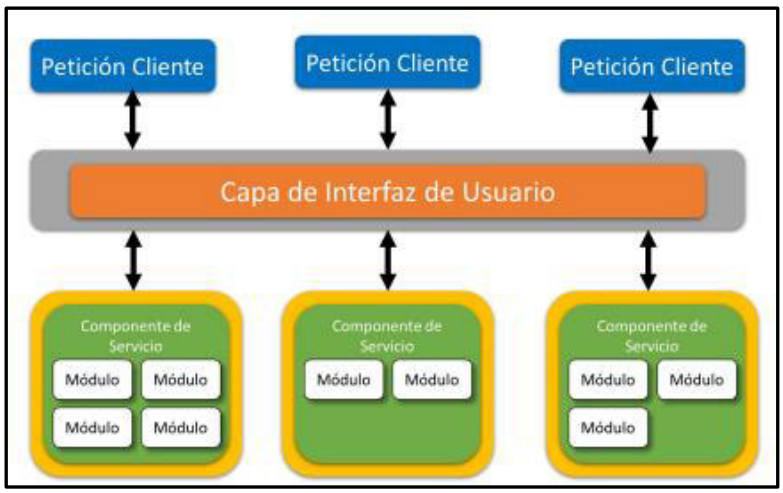



**Figura 10: Esquema de arquitectura microservicios**

**Nota: Adaptado de López y Maya** (2017)

### 3.2.4.17. Arquitectura Micro-Frontend

Según Mezzalira (2021)**,** los micro-frontends son una arquitectura emergente inspirada en la arquitectura de microservicios. Su principal idea es la descomponer el

código monolítico en pequeñas partes, esto para permitir a la organización distribuir el desarrollo a través de equipos independientes sin ralentizar la entrega del producto final.

En las siguientes figuras se presentan 2 escenarios para ver las diferencias entre la arquitectura web monolítica (ver figura 11) y la arquitectura micro-frontend (ver figura 12).

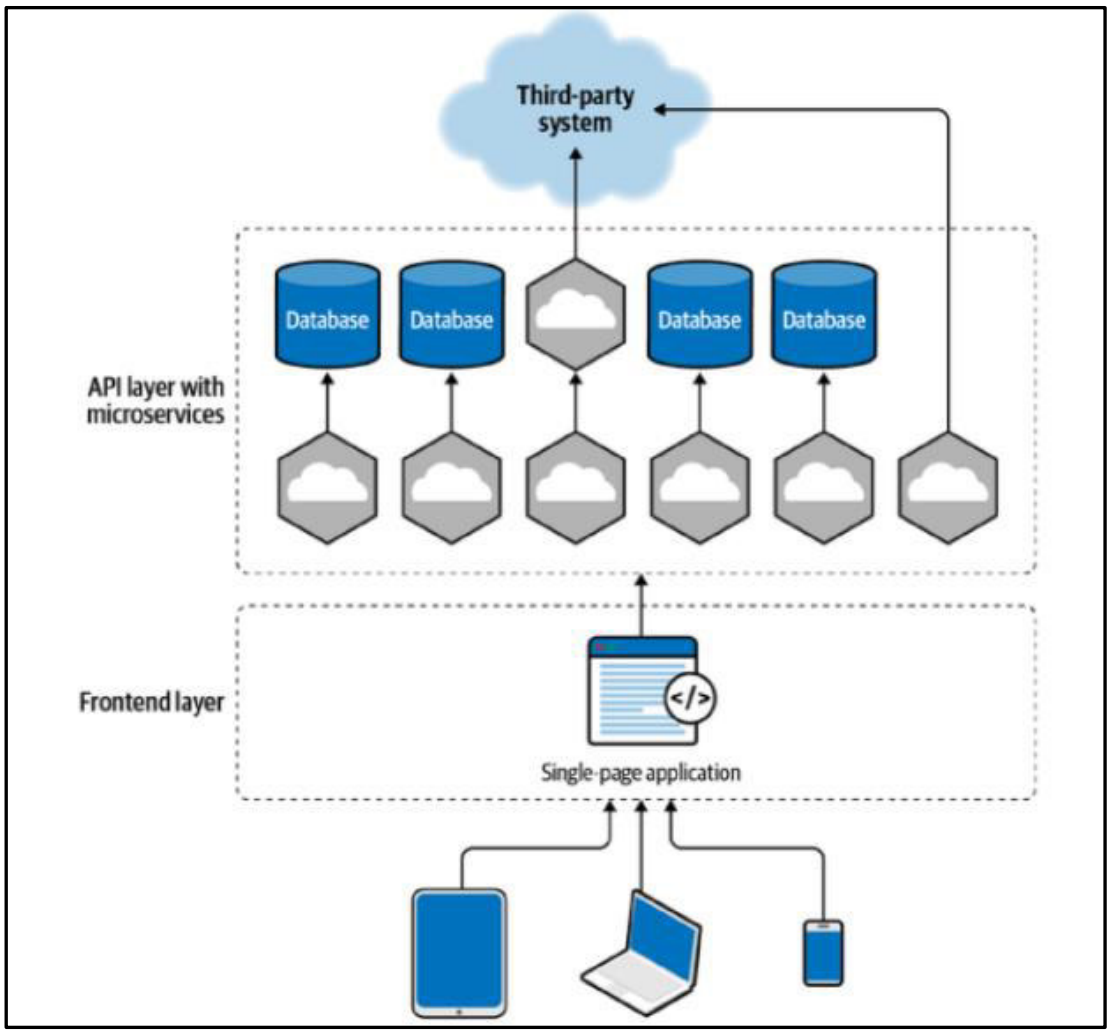



**Figura 11: Aplicación SPA interactuando con una arquitectura microservicios**

**Nota: Adaptado de (Mezzalira, 2021)**

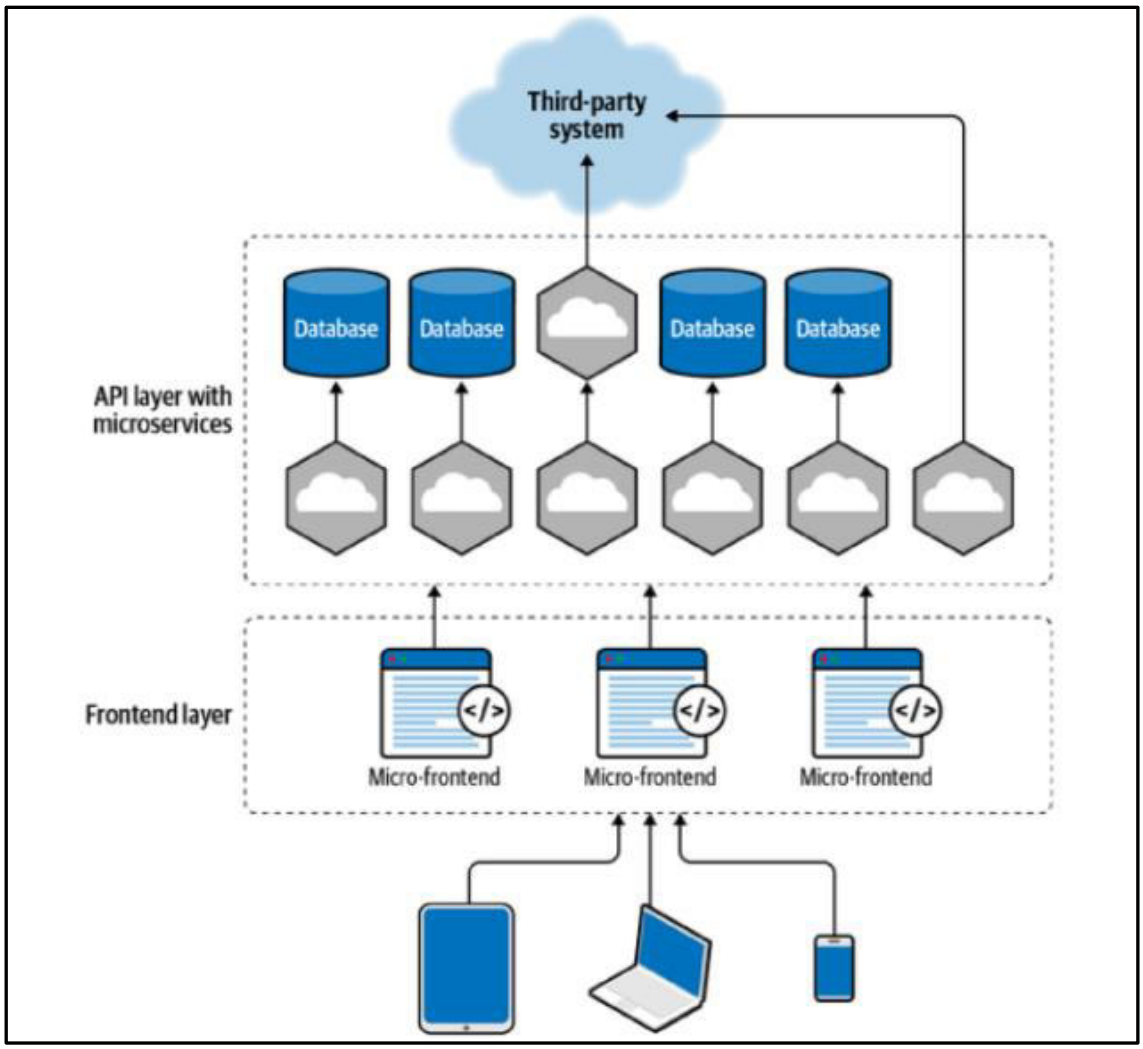



**Figura 12: Arquitectura micro-frontend interactuando con una arquitectura microservicios**

**Nota: Adaptado de (Mezzalira, 2021)**

### 3.2.5. Implementación de las áreas de procesos y sus buenas prácticas

De acuerdo a lo expuesto en el punto 3.2.3 Etapas y Metodologías, a continuación, se desarrollará el detalle de la implementación y los procesos realizados del presente proyecto, los cuales el autor se encargó de implementar.

#### 3.2.5.1. Definición del Product Backlog

En la siguiente tabla se presentan las actividades que forman parte del producto backlog del proyecto.

**Tabla 5: Product Backlog del proyecto**

| Item | Actividad |
|---|---|
| **1** | Elegir la tecnología adecuada para la implementación de la arquitectura micro-frontend y desacoplar los módulos de la aplicación web. |
| **2** | Limpiar y organizar el código fuente. |
| **3** | Realizar la migración de versión a Angular 12 de la librería de componentes web del SIU. |
| **4** | Realizar la migración de la aplicación web del SIU a la versión 12 del framework Angular. |
| **5** | Separar los módulos del SIU en sub-proyectos Angular e implementar dentro de ellos Webpack Module Federation. |
| **6** | Despliegue en ambiente de desarrollo y pase a producción. |

**Nota: Elaboración propia**

**Tabla 6: Actividades por sprint**

| Sprint | Evento | Actividad | Responsables |
|---|---|---|---|
| 1 | Sprint Planning | Definir cómo se implementará la arquitectura y plantear la necesidad de desacoplar los módulos de la aplicación web. | Product Owner<br>Scrum Master<br>Scrum Team |
| | Desarrollo del Sprint | Elegir la tecnología adecuada para la implementación de la arquitectura micro-frontend y desacoplar los módulos de la aplicación web. | Scrum Team |
| 2 | Sprint Planning | Plantear la necesidad de limpiar y organizar el código fuente, para eliminar posibles errores al implementar la nueva arquitectura. | Product Owner<br>Scrum Master<br>Scrum Team |
| | Desarrollo del Sprint | Limpiar y organizar el código fuente. | Scrum Team |
| 3 | Sprint Planning | Plantear la necesidad de migrar la librería de componentes web a la versión 12 del framework Angular. | Product Owner<br>Scrum Master<br>Scrum Team |
| | Desarrollo del Sprint | Realizar la migración de versión a Angular 12 de la librería de componentes web del SIU. | Scrum Team |
| 4 | Sprint Planning | Plantear la necesidad de migrar la aplicación web del SIU a la versión 12 del framework Angular. | Product Owner<br>Scrum Master<br>Scrum Team |
| | Desarrollo del Sprint | Realizar la migración de la aplicación web del SIU a la versión 12 del framework Angular. | Scrum Team |
| 5 | Sprint Planning | Plantear la necesidad de separar los módulos del SIU en sub proyectos Angular para implementar Webpack Module Federation. | Product Owner<br>Scrum Master<br>Scrum Team |
| | Desarrollo del Sprint | Separar los módulos del SIU en sub-proyectos Angular e implementar dentro de ellos Webpack Module Federation. | Scrum Team |
| 6 | Sprint Planning | La nueva arquitectura tiene que empezar a utilizarse y mostrar sus beneficios, por lo cual se realizará un despliegue completo de la aplicación web. | Product Owner<br>Scrum Master<br>Scrum Team |
| | Desarrollo del Sprint | Despliegue en ambiente de desarrollo y pase a producción. | Scrum Team |

**Nota: Elaboración propia**

#### 3.2.5.2. Sprint 1

**Sprint Planning**

Para proceder con la implementación de la arquitectura, se debe definir cual será la tecnología que permita la correcta implementación, asimismo es necesario eliminar uno de los principales obstáculos para la implementación de la nueva arquitectura, el código acoplado entre módulos. Para esto debemos asegurar que entre módulos de la aplicación web del SIU no exista dependencias entre estas y de haberlas, implementar la solución adecuada para su eliminación.

**Desarrollo del Sprint**

El autor del presente informe desarrolló el ítem N° 1 del Product Backlog: Elegir la tecnología adecuada para la implementación de la arquitectura micro-frontend y desacoplar los módulos de la aplicación web.

**Elección de la tecnología a utilizar**

En este punto el siguiente paso consistió en elegir la tecnología mediante la cual se integrarán las apps creadas y la aplicación host, sin que se vea afectada la experiencia de SPA que ya se tenía en producción. Se realizó una evaluación con 3 posibles tecnologías a usar.

- **Utilizar Iframes**

Los iframes podrían resultar la opción más sencilla. Se mostrarían las apps a través del iframe cargándolos según la necesidad de la aplicación.

Pese a que la idea resultó factible presentó las siguientes limitantes:

- No se puede compartir la sesión ya que, pese a que el módulo se carga dentro de la aplicación principal, esta sigue siendo una aplicación independiente y no existe manera de compartir información de la sesión de usuario entre la aplicación principal y los módulos independientes.
- La experiencia de usuario SPA se vería afectada, ya que perderíamos la posibilidad de acceder a una página de la aplicación directamente mediante un link directo, esto debido a que los iframes manejan sus rutas internamente.
- Riesgo de seguridad, ya que en el iframe puede ser vulnerable a ataques maliciosos.

- **Utilizar la librería single-spa**

Single Spa es un framework que permite juntar múltiples aplicaciones microfrontend javascript en una sola aplicación frontend. Este framework es una buena opción para implementar una arquitectura microfrontend en los proyectos, permite unificar microfrontends javascript desarrollados en distintos frameworks o librerías, sin embargo, el soporte que tiene para el framework Angular no es completo y existe el riesgo de presentar errores.

- **Utilizar Webpack Module Federation**

Webpack en su versión 5 nos presenta "Module Federation", esta característica nos permite cargar módulos, en nuestro caso, nuestras apps, dentro una aplicación contenedora de manera remota. Esta característica de webpack se ajusta más a la necesidad del proyecto.

Luego de evaluar las 3 opciones se eligió Webpack Module Federation como tecnología a utilizar debido a que webpack forma parte del framework Angular, esto permitirá que la implementación de la arquitectura sea soportada de manera nativa por el propio framework, lo cual disminuirá los riesgos de presentar posibles errores en la funcionalidad de la aplicación web si se implementara la librería single-spa.

**Desacoplar los módulos de la aplicación web**

Para entender el objetivo de esta actividad se explicará cómo está estructurada la aplicación web del SIU.

En la siguiente figura se muestra cómo está estructurada la aplicación.

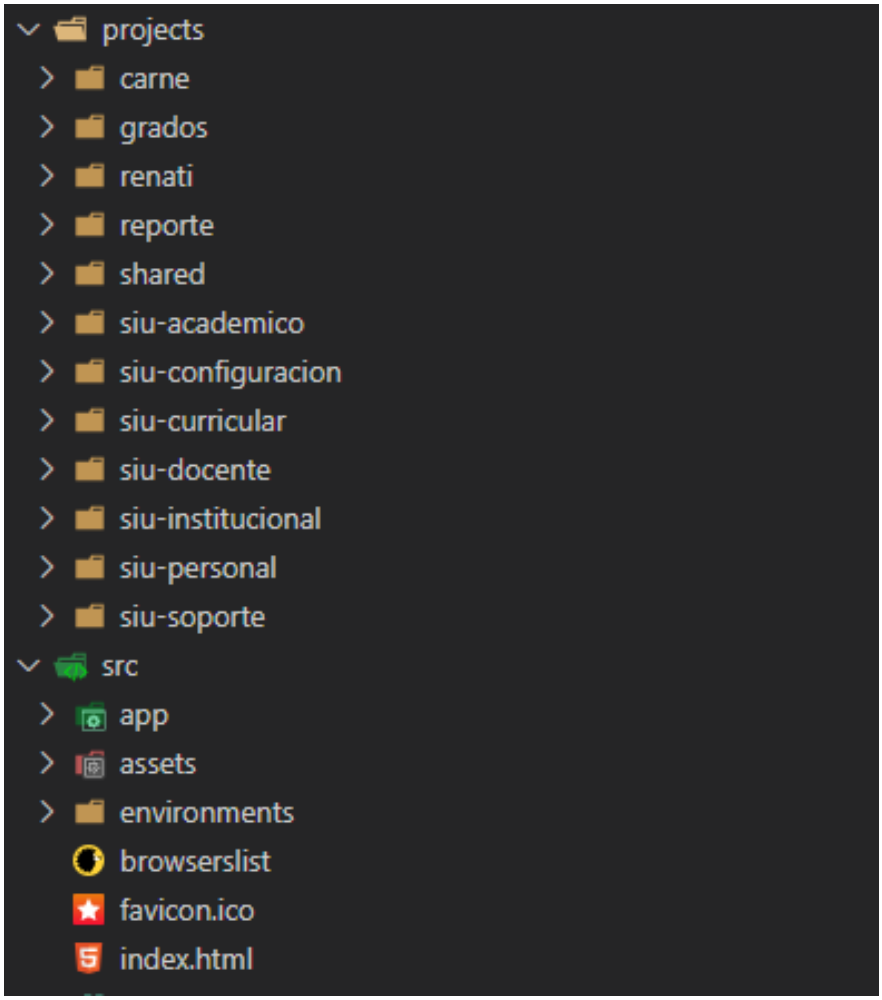



**Figura 13: Estructura monolítica de la aplicación web del SIU**

**Nota: Elaboración propia**

Dentro de la carpeta “projects” se encuentra el código fuente de los módulos del SIU así como de la primera versión de la librería de componentes web, la cual posteriormente se separó como un paquete de npm independiente pero en los módulos más antiguos se sigue referenciando a esta, y en la carpeta “src” los componentes core (aquellos que son reutilizados por todos los módulos y contienen lógica del negocio), páginas de login y errores, y el componente principal que toda aplicación desarrollada con el framework Angular tiene.

Para efectos de mejorar la importación de archivos dentro del proyecto, a cada módulo se le definió un espacio de nombres, el cual servirá para importar cualquier archivo de este módulo.

Angular permite definir en el archivo tsconfig.json un espacio de nombres que servirán como alias en la aplicación, en la figura 14 se presenta cómo están definidos los espacios de nombres en la aplicación web del SIU y en la figura 15 se presenta el uso de estos espacios de nombres.

```
"paths": {
  "@siu/academico/*": [
    "projects/siu-academico/src/*"
  ],
  "@siu/institucional/*": [
    "projects/siu-institucional/src/*"
  ],
  "@siu/configuracion/*": [
    "projects/siu-configuracion/src/*"
  ],
  "@siu/docente/*": [
    "projects/siu-docente/src/*"
  ],
  "@siu/soporte/*": [
    "projects/siu-soporte/src/*"
  ],
  "@siu/carne/*": [
    "projects/carne/src/*"
  ],
```

**Figura 14: Espacios de nombres configurados en la aplicación web del SIU**

**Nota: Elaboración propia**

```
1  import { IFormularioModel } from '@siu/grados/app/nacionales/interfaces';
2  import { Component, OnInit, Input } from '@angular/core';
3  import { EnviarNotificacionStore } from '../store/enviar-notificacion.store';
4  import {
5    IDataGridEvent,
6    IDataGridButtonEvent,
7    MapPageRequestFromGridSource,
8    AlertService,
9    ToastService,
10 } from '@sunedu/shared';
```

**Figura 15: Uso de los espacios de nombres en el código de la aplicación**

**Nota: Elaboración propia**

De acuerdo a la estructura del SIU que se acaba de explicar se puede entender el motivo por el que es necesario desacoplar los códigos de los módulos.

En las figuras 16 y 17 se presentan ejemplos del código acoplado entre módulos de la aplicación web del SIU.

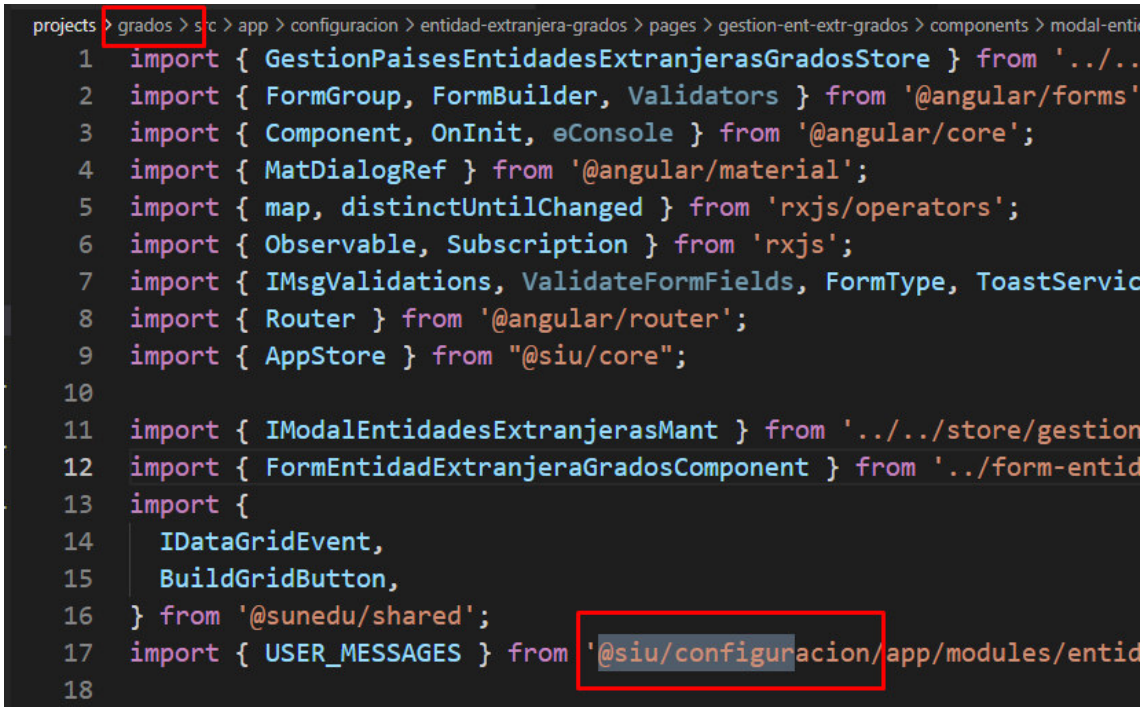



**Figura 16: Referencia incorrecta a el módulo de configuración en el módulo de grados**

**Nota: Elaboración propia**

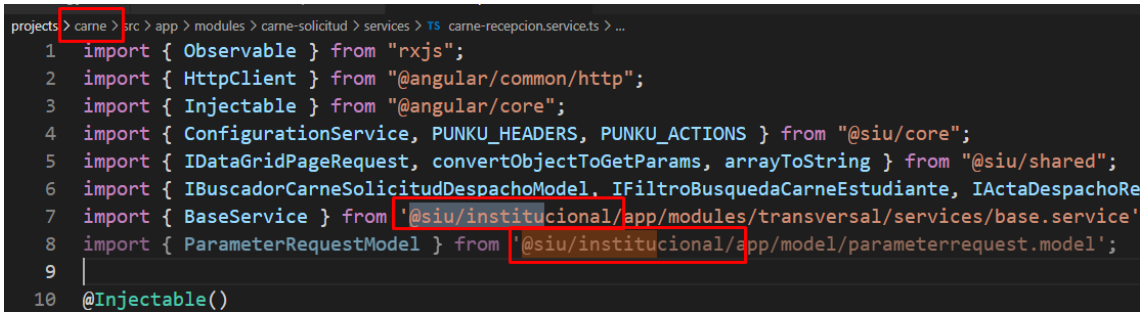



**Figura 17: Referencia incorrecta a el módulo institucional desde el módulo de carné**

**Nota: Elaboración propia**

El objetivo de este sprint fue buscar dentro de la aplicación y eliminar todas estas referencias incorrectas entre módulos, para permitir posteriormente compilar los módulos como sub proyectos independientes sin inconvenientes. Debido a la cantidad de

archivos que presenta la aplicación, esta tarea resulta laboriosa y entre algunas de las adecuaciones que se tuvo que realizar para llevar esta tarea a cabo fue la de recrear estos componentes que eran requeridos entre aplicaciones dentro de los componentes core de la aplicación web del SIU, de esta manera los módulos podrán acceder a componentes compartidos solamente a través de los componentes core.

En la siguiente figura se presenta un ejemplo de la recreación de un servicio que se encontraba en el módulo institucional y que era utilizado por otros proyectos como carné, grados, reportes y autoridades.

```
export class BaseHttpOldParameterRequestModel {
    Name: string;
    Value: any;

    constructor(name: string, value) {
        this.Name = name;
        this.Value = value;
    }
}

// servicio antiguo extraido de institucional
// se paso a core para desacoplar los proyectos
@Injectable({
    providedIn: 'root',
})
export class BaseHttpOldService {
    constructor(
        private httpClient: HttpClient,
        private config: ConfigurationService,
    ) { }

    /*Descarga de archivo de excel*/
    public async exportarExcel(
        urlService: string,
        parameters: HttpParams,
        addbaseUrlApiService: boolean = true,
    ): Promise<Blob> {
        let url = urlService;
        //let headers = this.setDefaultHeaders('CON');
        const file = await this.httpClient
            .get<Blob>(url, {
                params: parameters,
                headers: { [PUNKU_HEADERS.CODIGO_ACCION]: PUNKU_ACTIONS.CONSULTAR },
                responseType: 'blob' as 'json' /*, withCredentials: true */,
            })
```

**Figura 18: Recreación de un servicio del módulo institucional dentro de los componentes core del SIU**

**Nota: Elaboración propia**

Una vez realizado este procedimiento los módulos del SIU solamente podrán tener referencias hacia recursos del propio módulo, recursos de la librería de componentes web o de los componentes core de la aplicación.

En la figura 19 se presenta un esquema de cómo se encontraban las referencias hacia recursos en el código fuente de la aplicación web y en la figura 20 se presenta cómo quedaron luego de desacoplar los módulos.

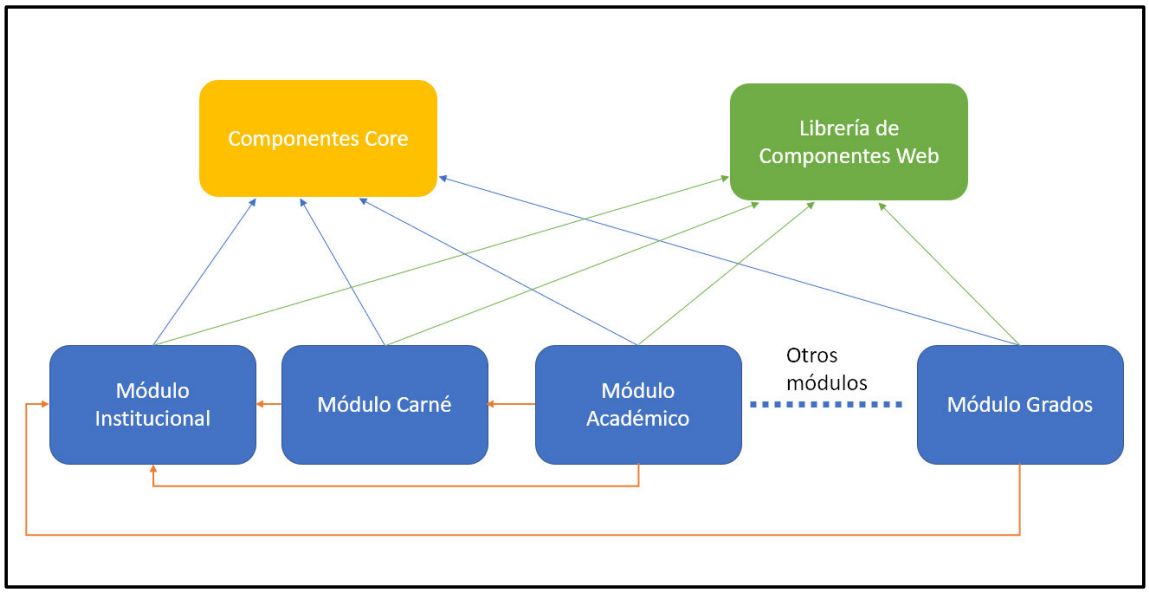



**Figura 19: Esquema de referencias entre módulos del SIU antes del Sprint 1**

**Nota: Elaboración propia**

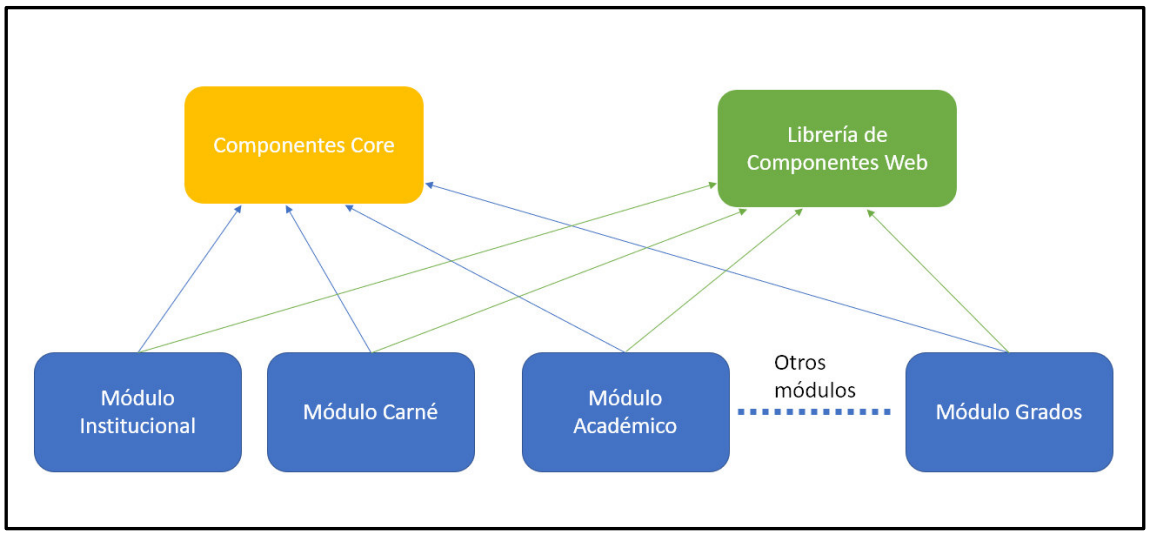



**Figura 20: Esquema de referencias entre módulos del SIU después del Sprint 1**

**Nota: Elaboración propia**

### 3.2.5.3. Sprint 2

**Sprint Planning**

Para mitigar otro potencial riesgo de errores al implementar la arquitectura micro-frontend en la aplicación web del SIU, se realizará una limpieza de código fuente eliminando referencias incorrectas hacia los componentes core, a la librería de componentes web o a referencias a recursos dentro del propio módulo.

**Desarrollo del Sprint**

El autor del presente informe desarrolló el ítem N° 2 del Product Backlog: Limpiar y organizar el código fuente.

Para entender el objetivo de esta actividad, se describirán ejemplos de referencias incorrectas en el proyecto. Básicamente existen 2 tipos de referencias incorrectas que se deben de corregir en todo el código de la aplicación. Estas son las siguientes:

- Referencias incorrectas hacia los componentes core
- Referencias incorrectas hacia la librería de componentes web

Sobre las referencias incorrectas hacia los componentes core, como se mencionó en el desarrollo del sprint anterior, los componentes core están ubicados en la carpeta "src", y para acceder a ellos se definió un espacio de nombres, "@siu/core", sin embargo, a lo largo de todo el código se observan referencias por ruta relativa hacia estos componentes, lo cual ocasionaría problemas al reestructurar el código.

En la figura 21 se presenta un ejemplo de este error y en la figura 22 se presenta la correcta referencia.

```
ojects > carne > src > app > modules > acta-entrega > pages > acta-entrega > components > forms > form-editar-acta > TS form-editar-acta.component.ts > ...
1  import { TipoRolSiuEnum } from './../../../../../../../../../../../../src/app/core/enums/siu-config.enum';
2  import { AppStore } from './../../../../../../../../../../../../src/app/core/store/app.store';
3  import { EstadoActaEntregaEnum } from './../../../../../../../../../../../../src/app/core/enums/siu-carne.enum';
4  import { GenericMessage } from './../../../../../../../../../../../shared/src/lib/services/toast.service';
5  import { MENSAJES } from './../../../../../../constantes/mensaje.constante';
6  import { IActaEntregaBuscadorModal } from './../../../../../../interfaces/acta-entrega-buscador-modal.interface
7  import { FormType } from './../../../../../../../../../../../shared/src/lib/enums/form.enum';
8  import { MatDialogRef } from '@angular/material';
9  import { AlertService } from './../../../../../../../../../../../shared/src/lib/services/alert.service';
10 import { IBuscadorModalActaEntrega } from './../../../../store/acta-entrega.store.interface';
11 import { FormModel, DEFAULT_MESSAGES, BuildGridButton, ToastService } from '@siu/shared';
12 import { Observable } from 'rxjs';
13 import { Component, OnInit } from '@angular/core';
14 import { ActaEntregaStore } from '../../../store/acta-entrega.store';
15 import { map, distinctUntilChanged } from 'rxjs/operators';
```

**Figura 21: Referencia incorrecta hacia los componentes core**

**Nota: Elaboración propia**

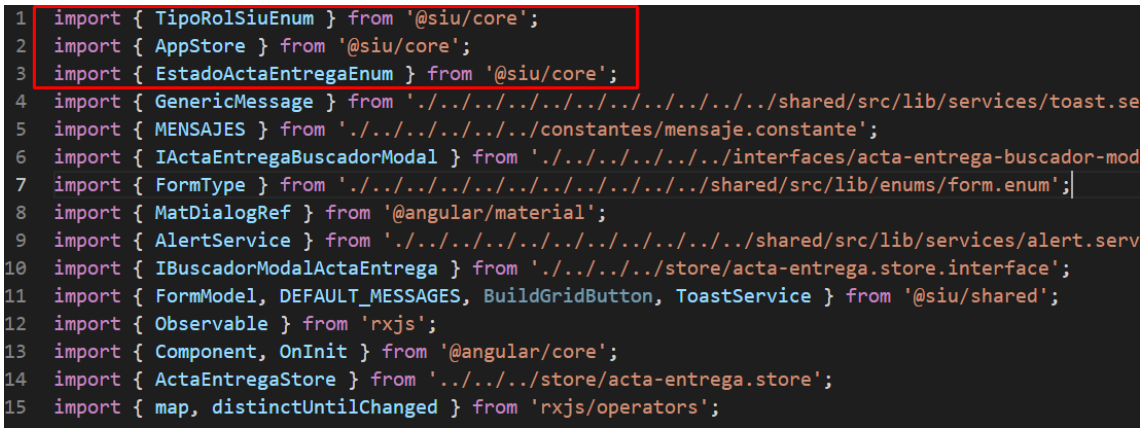



**Figura 22: Referencia correcta hacia los componentes core**

**Nota: Elaboración propia**

Para el ejemplo presentado en la figura anterior, al pertenecer las 2 primeras referencias al mismo espacio de nombres, se pueden unificar en una sola línea, quedando como se presenta en la siguiente figura.

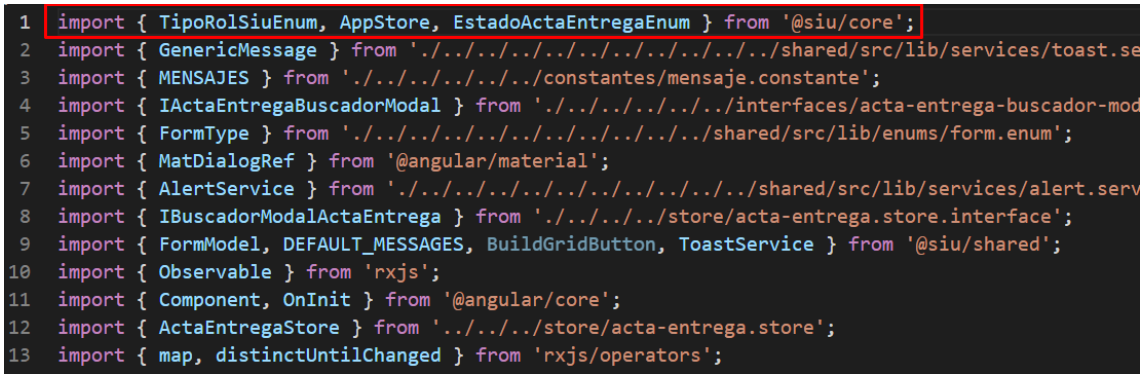
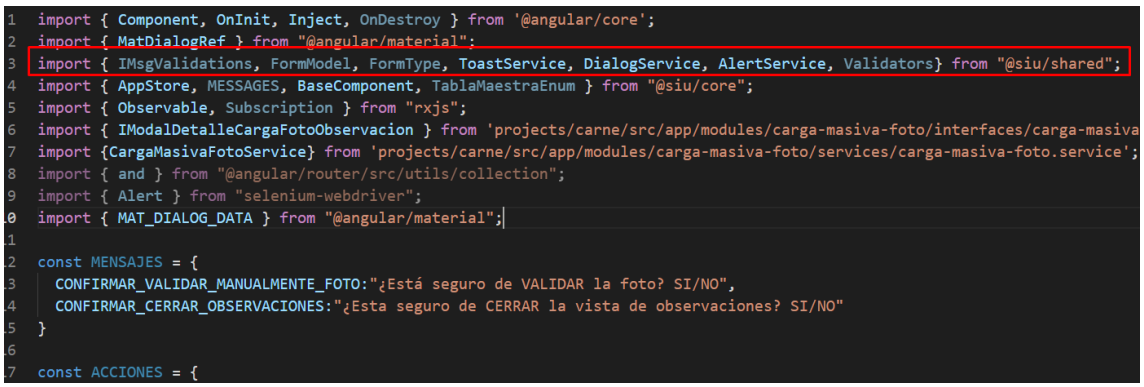



**Figura 23: Referencias correctas hacia los componentes core unificadas**

**Nota: Elaboración propia**

Sobre las referencias incorrectas hacia la librería de componentes web, se tiene definido un espacio de nombres hacia esta librería el cual se define como "@siu/shared", como se mencionó en el desarrollo del sprint anterior, en la aplicación web se tiene el código de la primera versión de esta librería la cual es utilizada en los módulos iniciales, para los módulos más recientes se creó un paquete npm independiente llamado "@sunedu/shared". Pese a esto a lo largo de todo el código de la aplicación se pueden observar referencias hacia @siu/shared y @sunedu/shared, incluso en los proyectos mas recientes. El trabajo en este caso es eliminar toda referencia, correcta o incorrecta de la primera versión de la librería de componentes y cambiarla a la versión más reciente, @sunedu/shared.

En las siguientes figuras se presentan ejemplos de las referencias hacia la librería de componentes web y el cambio hacia la referencia actual.



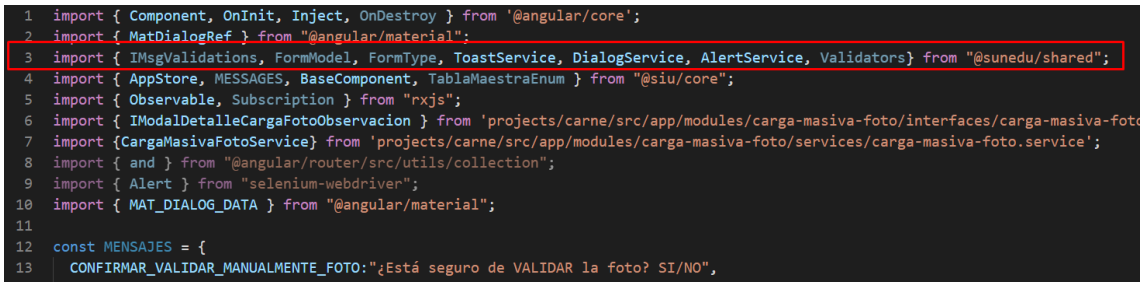
```
import { Component, OnInit, Inject, OnDestroy } from '@angular/core';
import { MatDialogRef } from "@angular/material";
import { IMsgValidations, FormModel, FormType, ToastService, DialogService, AlertService, Validators} from "@siu/shared";
import { AppStore, MESSAGES, BaseComponent, TablaMaestraEnum } from "@siu/core";
import { Observable, Subscription } from "rxjs";
import { IModalDetalleCargaFotoObservacion } from 'projects/carne/src/app/modules/carga-masiva-foto/interfaces/carga-masiva
import {CargaMasivaFotoService} from 'projects/carne/src/app/modules/carga-masiva-foto/services/carga-masiva-foto.service';
import { and } from "@angular/router/src/utils/collection";
import { Alert } from "selenium-webdriver";
import { MAT_DIALOG_DATA } from "@angular/material";

const MENSAJES = {
  CONFIRMAR_VALIDAR_MANUALMENTE_FOTO:"¿Está seguro de VALIDAR la foto? SI/NO",
  CONFIRMAR_CERRAR_OBSERVACIONES:"¿Esta seguro de CERRAR la vista de observaciones? SI/NO"
}

const ACCIONES = {
```

**Figura 24: Ejemplo de referencia hacia la primera versión de la librería de componentes web**

**Nota: Elaboración propia**

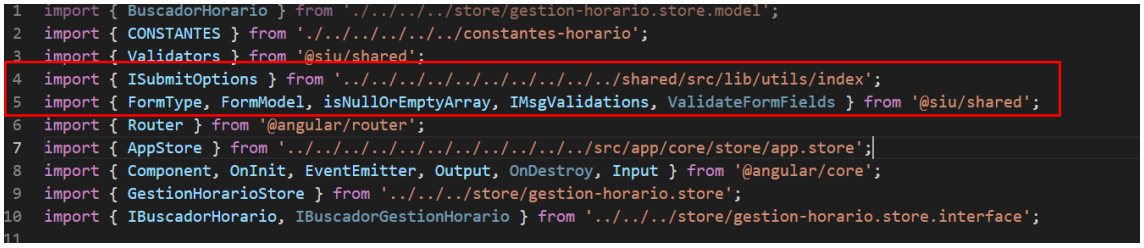


```
import { Component, OnInit, Inject, OnDestroy } from '@angular/core';
import { MatDialogRef } from "@angular/material";
import { IMsgValidations, FormModel, FormType, ToastService, DialogService, AlertService, Validators} from "@sunedu/shared";
import { AppStore, MESSAGES, BaseComponent, TablaMaestraEnum } from "@siu/core";
import { Observable, Subscription } from "rxjs";
import { IModalDetalleCargaFotoObservacion } from 'projects/carne/src/app/modules/carga-masiva-foto/interfaces/carga-masiva-foto
import {CargaMasivaFotoService} from 'projects/carne/src/app/modules/carga-masiva-foto/services/carga-masiva-foto.service';
import { and } from "@angular/router/src/utils/collection";
import { Alert } from "selenium-webdriver";
import { MAT_DIALOG_DATA } from "@angular/material";

const MENSAJES = {
  CONFIRMAR_VALIDAR_MANUALMENTE_FOTO:"¿Está seguro de VALIDAR la foto? SI/NO",
```

**Figura 25: Ejemplo de corrección de la referencia hacia la librería actual de componentes web**

**Nota: Elaboración propia**



```
import { BuscadorHorario } from '../../../../store/gestion-horario.store.model';
import { CONSTANTES } from './../../../../../../constantes-horario';
import { Validators } from '@siu/shared';
import { ISubmitOptions } from '../../../../../../../../../../../../shared/src/lib/utils/index';
import { FormType, FormModel, isNullOrEmptyArray, IMsgValidations, ValidateFormFields } from '@siu/shared';
import { Router } from '@angular/router';
import { AppStore } from '../../../../../../../../../../../../src/app/core/store/app.store';
import { Component, OnInit, EventEmitter, Output, OnDestroy, Input } from '@angular/core';
import { GestionHorarioStore } from '../../../store/gestion-horario.store';
import { IBuscadorHorario, IBuscadorGestionHorario } from '../../../store/gestion-horario.store.interface';
```

**Figura 26: Ejemplo de referencias correctas e incorrectas hacia la primera versión de la librería de componentes web**

**Nota: Elaboración propia**

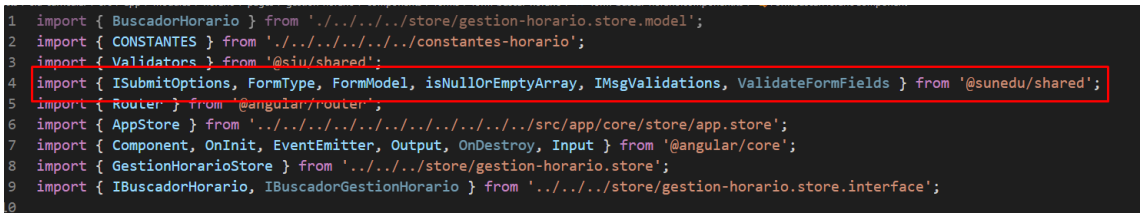



**Figura 27: Corrección y unificación hacia la versión actual de la librería de componentes web**

**Nota: Elaboración propia**

Se tuvo que realizar el proceso presentado en las figuras anteriores a lo largo de todo el código de la aplicación web del SIU, eliminando por completo toda referencia hacia la primera versión de la librería de componentes web. Para culminar esta tarea se debe cambiar en los archivos html los elementos correspondientes a la primera versión de la librería, todos aquellos componentes que empiecen con el prefijo "<siu-" al prefijo "<sunedu-", esto para que sea compatible con la nueva versión de la librería.

Esta tarea se realizó con mucha precaución ya que hubo demasiados componentes correspondientes a la primera versión de la librería de componentes web. Se encontraron 4427 casos de este tipo en el código de la aplicación web. En la siguiente figura se presenta la cantidad de ocurrencias encontradas en el código.

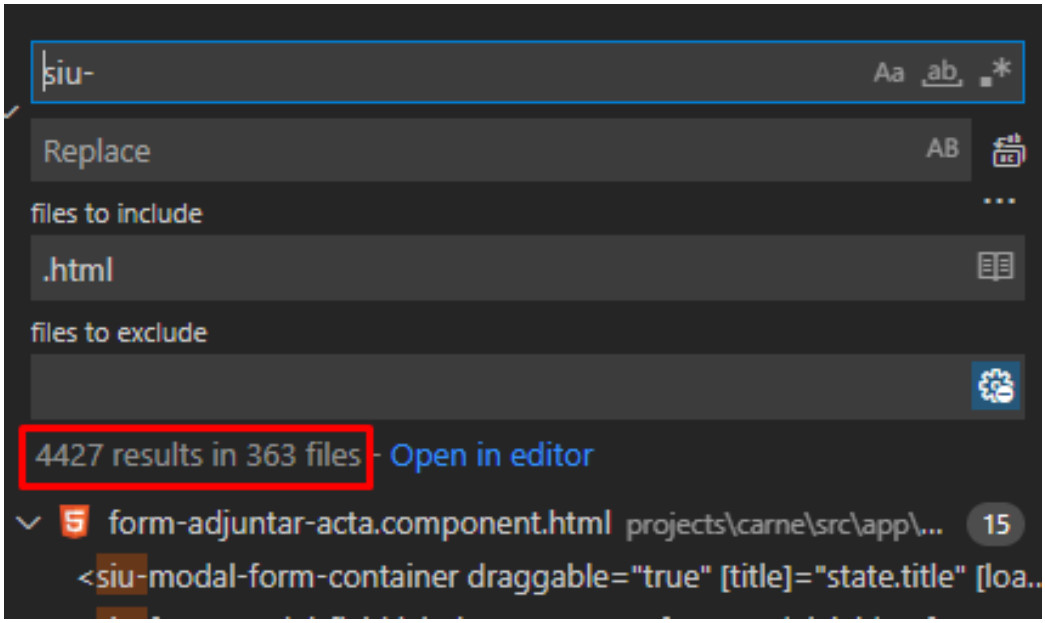
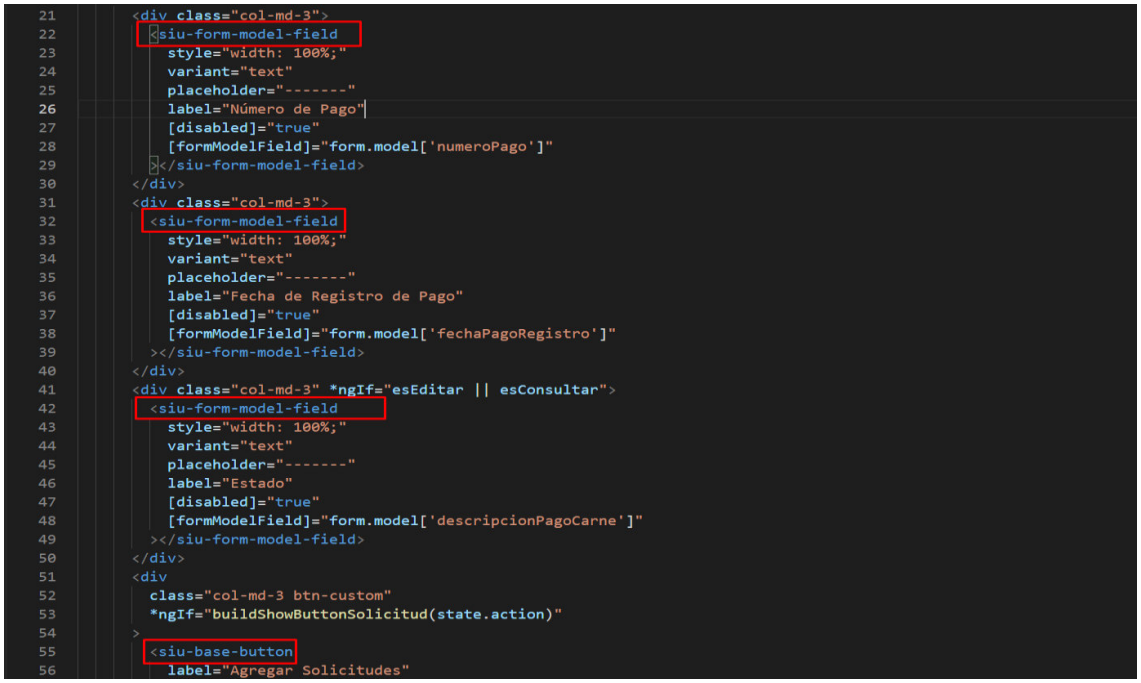



**Figura 28: Cantidad de ocurrencias encontradas correspondientes a la primera versión de la librería de componentes**

**Nota: Elaboración propia**

En la siguiente figura se observa un ejemplo de estos componentes.



```
<div class="col-md-3">
  <siu-form-model-field
    style="width: 100%;"
    variant="text"
    placeholder="-------"
    label="Número de Pago"
    [disabled]="true"
    [formModelField]="form.model['numeroPago']"
  ></siu-form-model-field>
</div>
<div class="col-md-3">
  <siu-form-model-field
    style="width: 100%;"
    variant="text"
    placeholder="-------"
    label="Fecha de Registro de Pago"
    [disabled]="true"
    [formModelField]="form.model['fechaPagoRegistro']"
  ></siu-form-model-field>
</div>
<div class="col-md-3" *ngIf="esEditar || esConsultar">
  <siu-form-model-field
    style="width: 100%;"
    variant="text"
    placeholder="-------"
    label="Estado"
    [disabled]="true"
    [formModelField]="form.model['descripcionPagoCarne']"
  ></siu-form-model-field>
</div>
<div
  class="col-md-3 btn-custom"
  *ngIf="buildShowButtonSolicitud(state.action)"
>
  <siu-base-button
    label="Agregar Solicitudes"
```

**Figura 29: Componentes web correspondientes a la primera versión de la librería**

**Nota: Elaboración propia**

En la siguiente figura se presenta el mismo html con el cambio realizado.

```
<div class="col-md-3">
  <sunedu-form-model-field
    style="width: 100%;"
    variant="text"
    placeholder="-------"
    label="Número de Pago"
    [disabled]="true"
    [formModelField]="form.model['numeroPago']"
  ></sunedu-form-model-field>
</div>
<div class="col-md-3">
  <sunedu-form-model-field
    style="width: 100%;"
    variant="text"
    placeholder="-------"
    label="Fecha de Registro de Pago"
    [disabled]="true"
    [formModelField]="form.model['fechaPagoRegistro']"
  ></sunedu-form-model-field>
</div>
<div class="col-md-3" *ngIf="esEditar || esConsultar">
  <sunedu-form-model-field
    style="width: 100%;"
    variant="text"
    placeholder="-------"
    label="Estado"
    [disabled]="true"
    [formModelField]="form.model['descripcionPagoCarne']"
  ></sunedu-form-model-field>
</div>
<div
  class="col-md-3 btn-custom"
  *ngIf="buildShowButtonSolicitud(state.action)"
>
  <sunedu-base-button
    label="Agregar Solicitudes"
```

**Figura 30: Cambio del prefijo de los componentes de la librería para que sean compatibles con la versión actual**

**Nota: Elaboración propia**

Corrigiendo estos problemas, nos aseguramos que al reestructurar la aplicación a la nueva arquitectura no presentará problemas de referencias o recursos no encontrados en el directorio.

### 3.2.5.4. Sprint 3

#### Sprint Planning

La aplicación web del SIU está desarrollada con el framework Angular versión 7, esta versión trabaja con webpack 4 la cual no tiene soporte para implementar la tecnología Webpack Module Federation.

Para lograr la implementación de la arquitectura de micro-frontend con la tecnología Webpack Module Federation es necesario migrar la aplicación web a la versión 12 del framework Angular, la cual incluye la versión 5 de webpack, versión que sí soporta Webpack Module Federation, que permitirá conseguir la independencia de los

módulos de la aplicación web sin sacrificar la experiencia SPA que brinda el framework Angular.

Previo a migrar la aplicación web del SIU, se debe migrar la librería de componentes web, que se utiliza como un paquete independiente, y que al igual que la aplicación web del SIU, está desarrollada con la versión 7 del framework Angular

**Desarrollo del Sprint**

El autor del presente informe desarrolló el ítem N° 3 del Product Backlog: Realizar la migración de versión a Angular 12 de la librería de componentes web del SIU código fuente.

Para migrar la versión de Angular a una versión superior, el procedimiento a realizar es incrementar la versión de manera consecutiva, es decir, si nos encontramos en la versión 7 y queremos llegar a la versión 12, debemos primero migrar a la versión 8, luego a la versión 9, y repetir el proceso hasta llegar a la versión 12. Al terminar la cada migración incremental y asegurarse de que compile adecuadamente se fueron realizando commits al controlador de versiones para que ante un posible error fuera sencillo hacer el rollback a una versión anterior.

Angular ofrece una guía para realizar la migración incremental por cada versión.

En la siguiente figura se presenta el contenido de esta guía

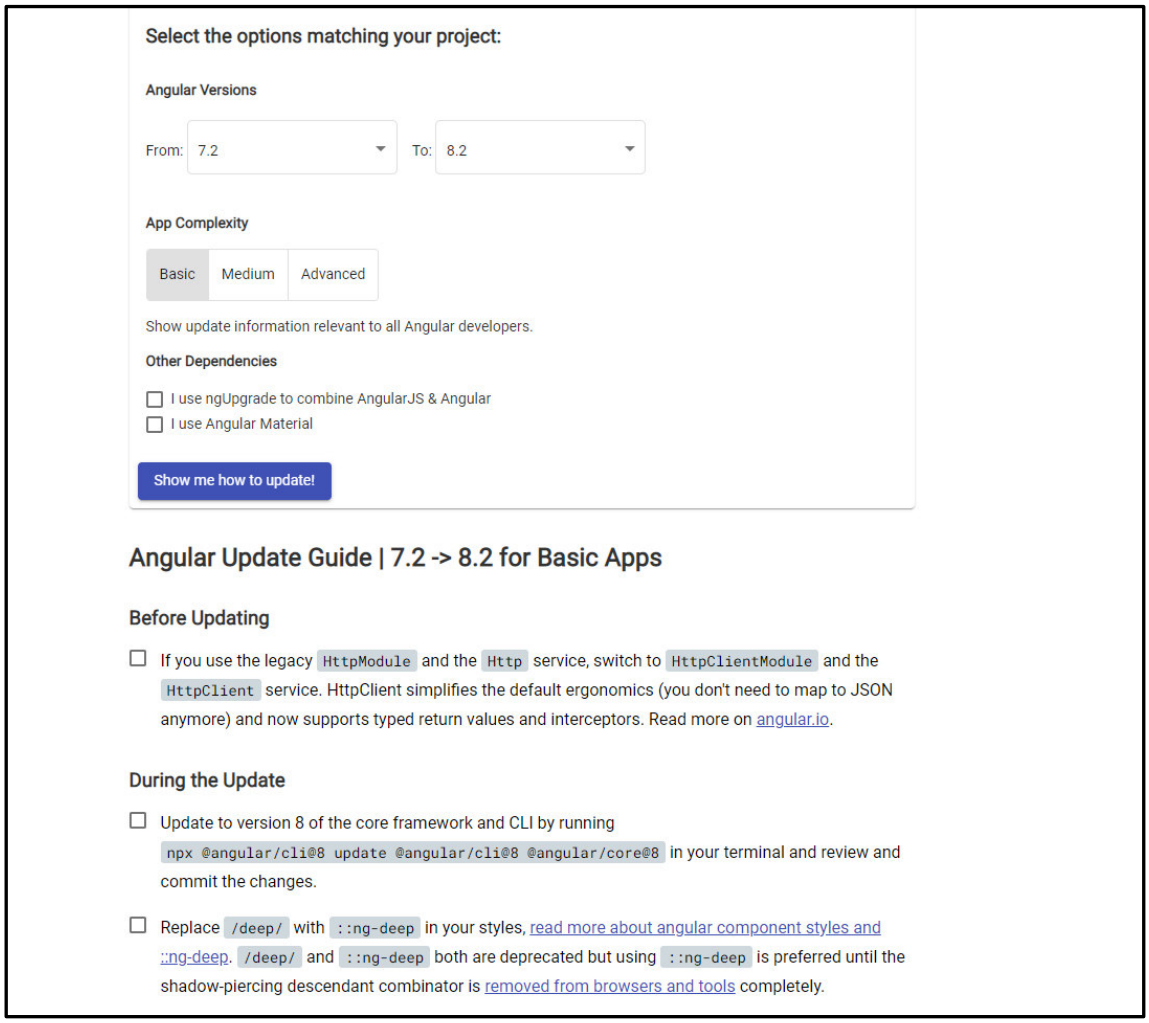
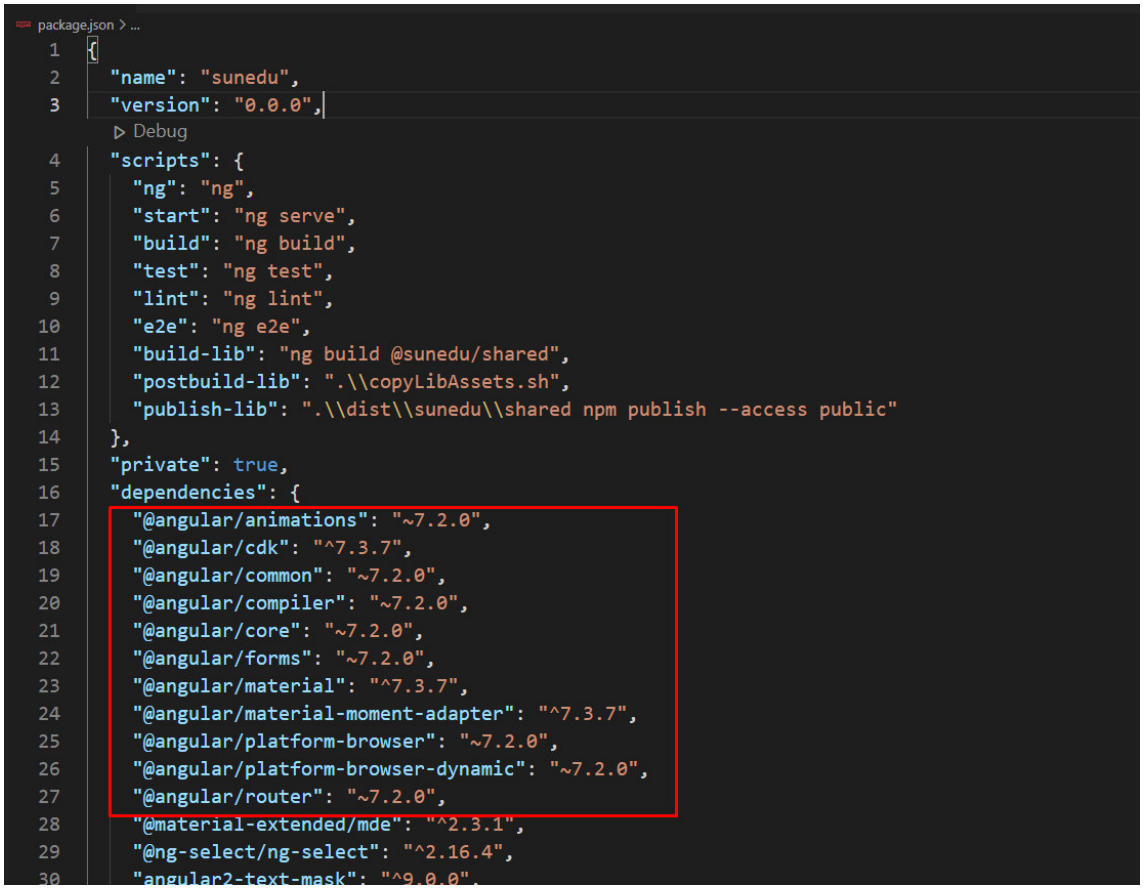



**Figura 31: Guía de migración de código fuente del Framework Angular**

**Nota: Adaptado de Angular Update Guide** (2021)

En la siguiente figura se presenta el archivo package,json en donde se muestra la versión en la cual se encuentra inicialmente el proyecto de la librería de componentes web del SIU.



```
package.json > ...
{
  "name": "sunedu",
  "version": "0.0.0",
  ▷ Debug
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e",
    "build-lib": "ng build @sunedu/shared",
    "postbuild-lib": ".\\copyLibAssets.sh",
    "publish-lib": ".\\dist\\sunedu\\shared npm publish --access public"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "~7.2.0",
    "@angular/cdk": "^7.3.7",
    "@angular/common": "~7.2.0",
    "@angular/compiler": "~7.2.0",
    "@angular/core": "~7.2.0",
    "@angular/forms": "~7.2.0",
    "@angular/material": "^7.3.7",
    "@angular/material-moment-adapter": "^7.3.7",
    "@angular/platform-browser": "~7.2.0",
    "@angular/platform-browser-dynamic": "~7.2.0",
    "@angular/router": "~7.2.0",
    "@material-extended/mde": "^2.3.1",
    "@ng-select/ng-select": "^2.16.4",
    "angular2-text-mask": "^9.0.0",
```

**Figura 32: Archivo package.json con la versión 7 de Angular en la librería de componentes**

**Nota: Elaboración propia**

Se procedió a realizar la migración del código fuente desde la versión 7.2 de Angular hasta la versión 8.2 según lo indicado en la guía de migración publicada por Angular.

https://update.angular.io/?l=2&v=7.2-8.2

Se procedió a realizar la migración del código fuente desde la versión 8.2 de Angular hasta la versión 9.1 según lo indicado en la guía de migración publicada por Angular.

https://update.angular.io/?l=2&v=8.2-9.1

Se procedió a realizar la migración del código fuente desde la versión 9.1 de Angular hasta la versión 10.2 según lo indicado en la guía de migración publicada por Angular.

https://update.angular.io/?l=2&v=9.1-10.2

Se procedió a realizar la migración del código fuente desde la versión 10.2 de Angular hasta la versión 11.0 según lo indicado en la guía de migración publicada por Angular.

https://update.angular.io/?l=2&v=10.2-11.0

Se procedió a realizar la migración del código fuente desde la versión 11.0 de Angular hasta la versión 12.0 según lo indicado en la guía de migración publicada por Angular.

https://update.angular.io/?l=2&v=11.0-12.0

Luego realizar las guías de migración y encontrarse en la versión 12 de Angular, el archivo package.json actualizado del proyecto se presenta en la siguiente figura donde se aprecian las dependencias del framework Angular en su versión 12.2.6.

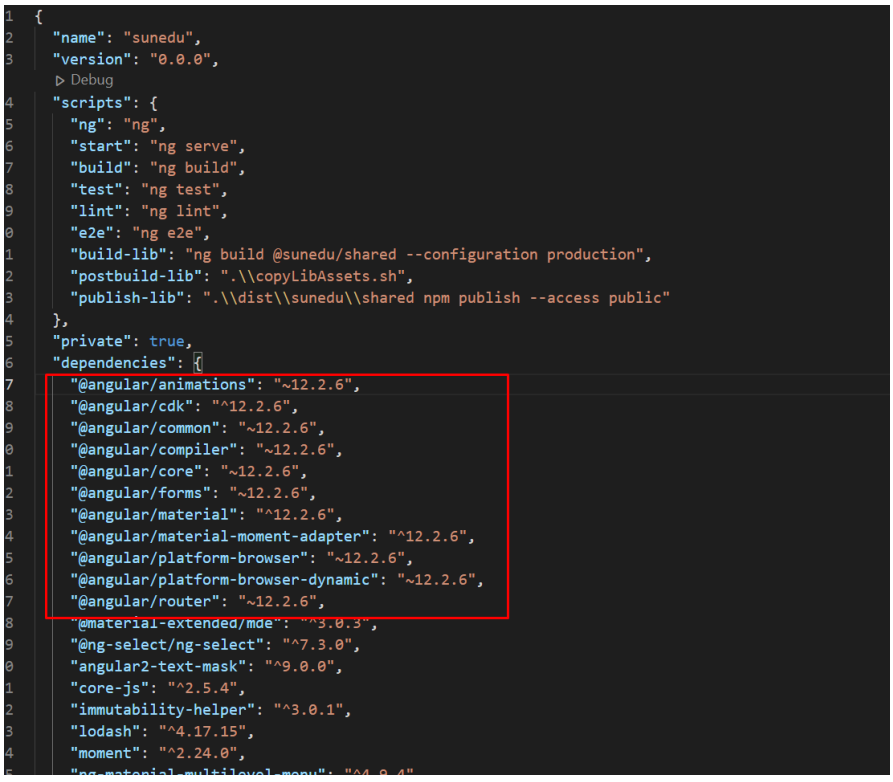



**Figura 33: Archivo package.json con la versión 12 de Angular en la librería de componentes**

**Nota: Elaboración propia**

Una vez realizada la migración se procedió a compilar y publicar la librería en el repositorio de npm para poder ser usada por la aplicación web del SIU cuando esta se migre a la versión 12 de Angular.

#### 3.2.5.5. Sprint 4

**Sprint Planning**

Como se explicó en el sprint anterior, es necesario migrar la aplicación a la versión 12 de Angular y así aplicar la tecnología de Webpack Module Federation, por lo tanto, se debe realizar la migración de toda la aplicación web del SIU a la versión 12 del framework Angular asegurándose que ninguna funcionalidad desarrollada presente errores luego de la migración.

**Desarrollo del Sprint**

El autor del presente informe desarrolló el ítem N° 4 del Product Backlog: Realizar la migración de la aplicación web del SIU a la versión 12 del framework Angular. Para realizar esta migración se emplearon los mismos pasos que en sprint anterior pero aplicado a la aplicación web del SIU los cuales se detallan a continuación.

Se procedió a realizar la migración del código fuente de la aplicación web del SIU desde la versión 7.2 de Angular hasta la versión 8.2 según lo indicado en la guía de migración publicada por Angular.

https://update.angular.io/?l=2&v=7.2-8.2

Se procedió a realizar la migración del código fuente de la aplicación web del SIU desde la versión 8.2 de Angular hasta la versión 9.1 según lo indicado en la guía de migración publicada por Angular.

https://update.angular.io/?l=2&v=8.2-9.1

Se procedió a realizar la migración del código fuente de la aplicación web del SIU desde la versión 9.1 de Angular hasta la versión 10.2 según lo indicado en la guía de migración publicada por Angular.

https://update.angular.io/?l=2&v=9.1-10.2

Se procedió a realizar la migración del código fuente de la aplicación web del SIU desde la versión 10.2 de Angular hasta la versión 11.0 según lo indicado en la guía de migración publicada por Angular.

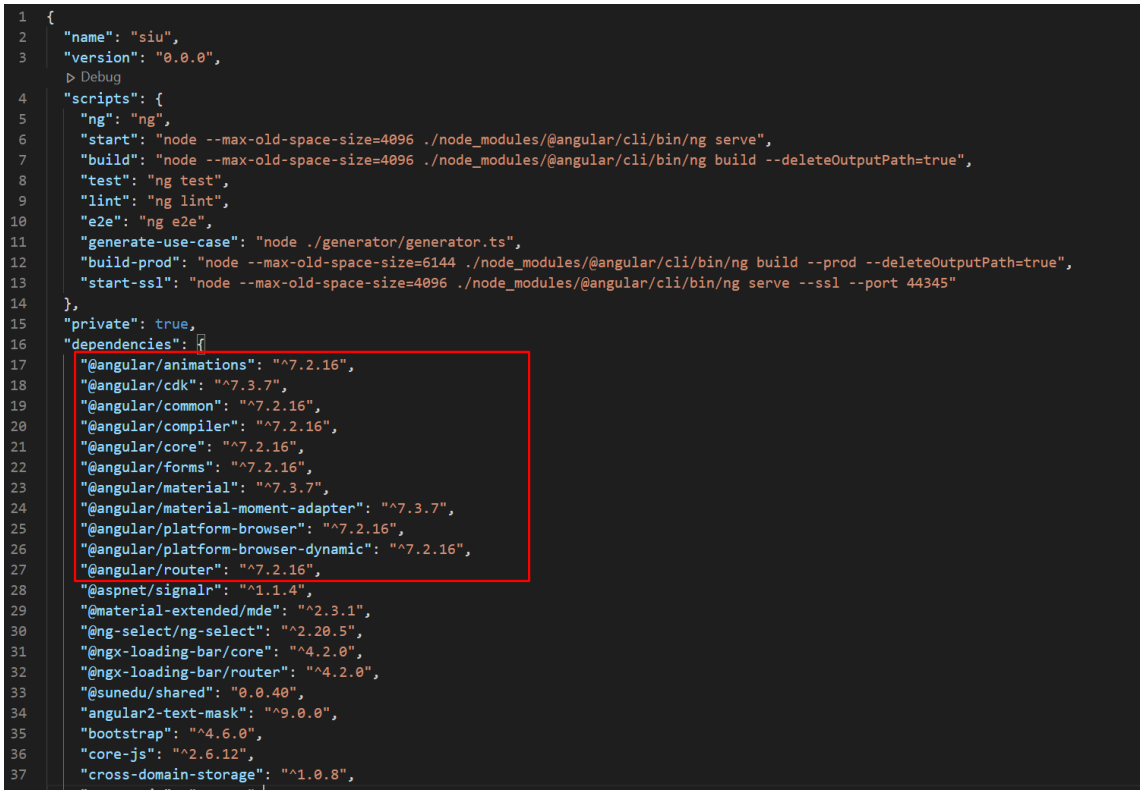
https://update.angular.io/?l=2&v=10.2-11.0

Se procedió a realizar la migración del código fuente de la aplicación web del SIU desde la versión 11.0 de Angular hasta la versión 12.0 según lo indicado en la guía de migración publicada por Angular.

https://update.angular.io/?l=2&v=11.0-12.0

Por cada incremento de versión se realizó la tarea de compilar el proyecto y desplegarlo en el ambiente de desarrollo en busca de posibles errores o incompatibilidades en la aplicación y de ser encontrados, se fueron corrigiendo uno por uno.

En las siguientes figuras se muestran el cambio que sufrió el archivo package JSON sobre las versiones de los paquetes del framework Angular.



```
{
  "name": "siu",
  "version": "0.0.0",
  ▷ Debug
  "scripts": {
    "ng": "ng",
    "start": "node --max-old-space-size=4096 ./node_modules/@angular/cli/bin/ng serve",
    "build": "node --max-old-space-size=4096 ./node_modules/@angular/cli/bin/ng build --deleteOutputPath=true",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e",
    "generate-use-case": "node ./generator/generator.ts",
    "build-prod": "node --max-old-space-size=6144 ./node_modules/@angular/cli/bin/ng build --prod --deleteOutputPath=true",
    "start-ssl": "node --max-old-space-size=4096 ./node_modules/@angular/cli/bin/ng serve --ssl --port 44345"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "^7.2.16",
    "@angular/cdk": "^7.3.7",
    "@angular/common": "^7.2.16",
    "@angular/compiler": "^7.2.16",
    "@angular/core": "^7.2.16",
    "@angular/forms": "^7.2.16",
    "@angular/material": "^7.3.7",
    "@angular/material-moment-adapter": "^7.3.7",
    "@angular/platform-browser": "^7.2.16",
    "@angular/platform-browser-dynamic": "^7.2.16",
    "@angular/router": "^7.2.16",
    "@aspnet/signalr": "^1.1.4",
    "@material-extended/mde": "^2.3.1",
    "@ng-select/ng-select": "^2.20.5",
    "@ngx-loading-bar/core": "^4.2.0",
    "@ngx-loading-bar/router": "^4.2.0",
    "@sunedu/shared": "0.0.40",
    "angular2-text-mask": "^9.0.0",
    "bootstrap": "^4.6.0",
    "core-js": "^2.6.12",
    "cross-domain-storage": "^1.0.8",
```

**Figura 34: Archivo package.json de la aplicación web del SIU antes de la migración**

**Nota: Elaboración propia**

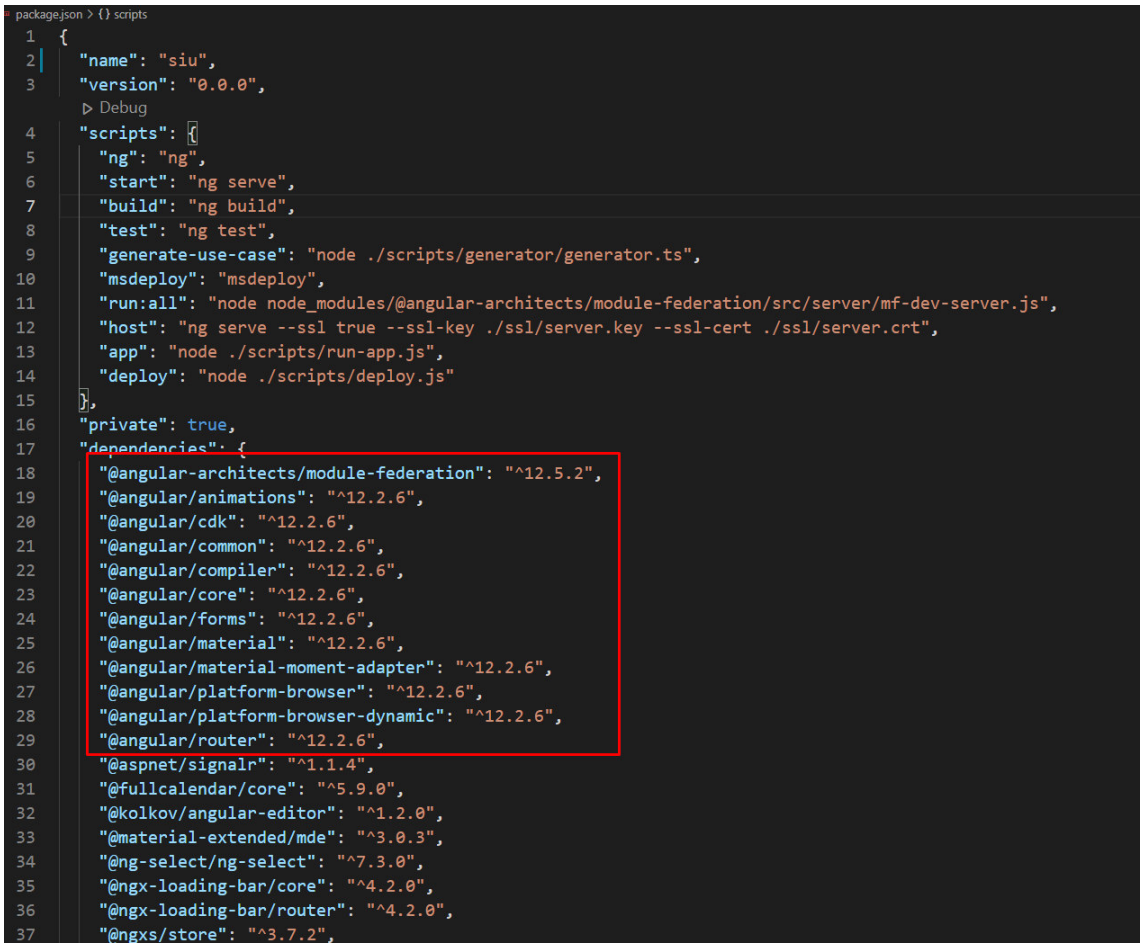



**Figura 35: Archivo package.json de la aplicación web del SIU después de la migración a Angular 12**

**Nota: Elaboración propia**

Una vez finalizado este sprint, se cuenta con un producto con código limpio respecto a referencias de recursos dentro de la aplicación, sin acoplamiento de código entre módulos y con la última versión estable, a la fecha de desarrollo del proyecto, del framework Angular la cual brindará mejor estabilidad hacia una arquitectura micro-frontend.

#### 3.2.5.6. Sprint 5

**Sprint Planning**

Para implementar el concepto de la arquitectura de micro-frontend se debe conseguir la independización de los módulos, para su mejor implementación y despliegue. Por esta razón se debe separar el código de los módulos de la aplicación web del SIU en pequeños proyectos Angular.

**Desarrollo del Sprint**

El autor del presente informe desarrolló el ítem N° 5 del Product Backlog: Separar los módulos del SIU en sub-proyectos Angular e implementar dentro de ellos Webpack Module Federation.

El objetivo de este sprint es aplicar dentro del proyecto web el concepto de micro-frontend y tener los módulos, que ahora serán llamados "apps", desplegados de manera independiente. Esta independencia llevará a cumplir los objetivos de reducción del tiempo de compilación ya que cada app individual demorará menos tiempo en compilar que todo el proyecto junto y a su vez al ser un despliegue por app, se eliminará los problemas de integración de código ya que cada app se alojará en su propio site.

Luego de la separación el proyecto SIU debe tener el esquema de arquitectura presentado en la siguiente figura, en la cual las apps o mircro-frontends (MF) se comunican con el Api Gateway hacia los microservicios que gestionan la lógica correspondiente a cada una de estas apps.

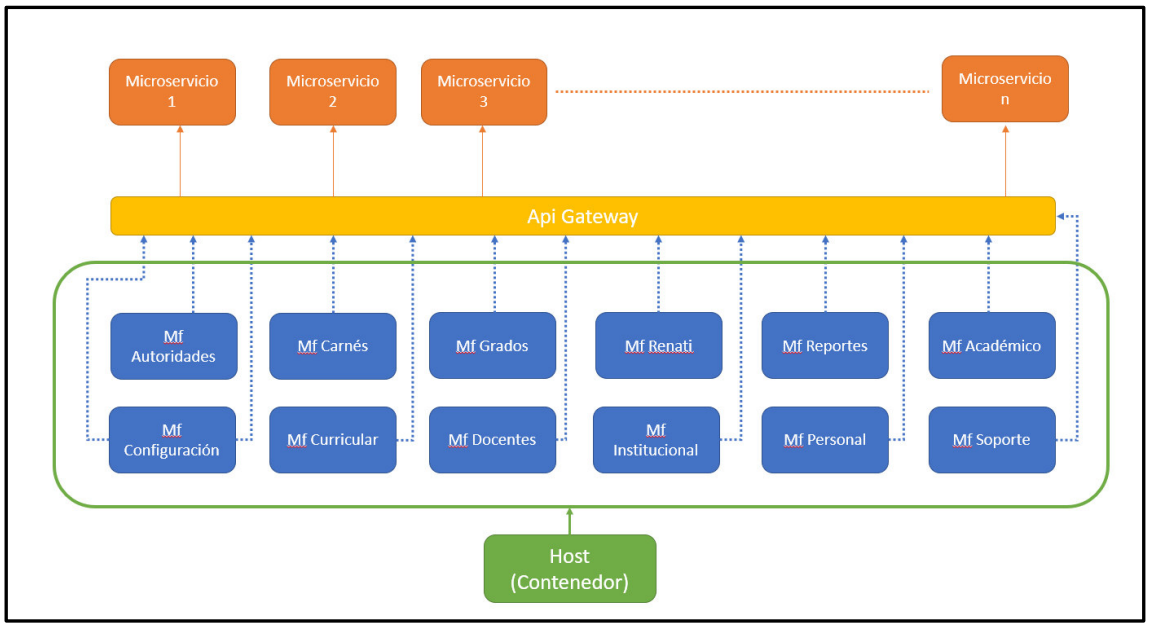



**Figura 36: Esquema de arquitectura micro-frontend aplicada en el SIU**

**Nota: Elaboración propia**

Se procedió a crear una carpeta apps dentro del proyecto, que será la que alojará los sub-proyectos de la aplicación web del SIU.

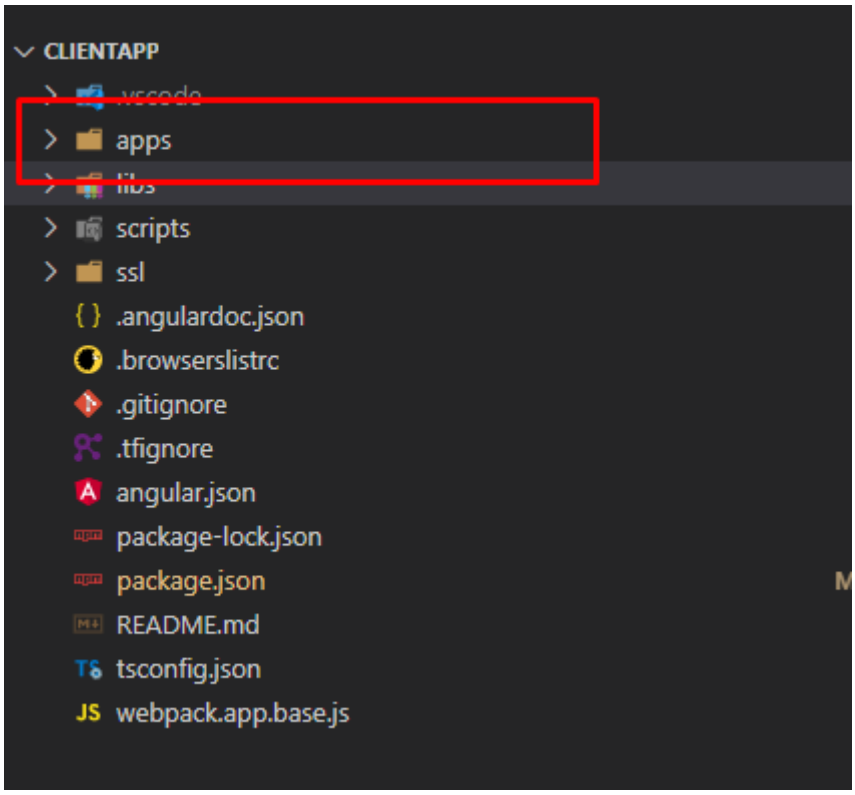



**Figura 37: Agregando carpeta apps en el proyecto web del SIU**

**Nota: Elaboración propia**

Se procedió a crear por cada módulo del SIU una aplicación angular dentro de la carpeta apps con el comando “ng generate application <nombre-app>”.


**Figura 38: Ejemplo de comando para generar aplicaciones en el SIU**

**Nota: Elaboración propia**

En la siguiente figura se presenta la carpeta apps con todos los proyectos creados para los módulos del SIU.

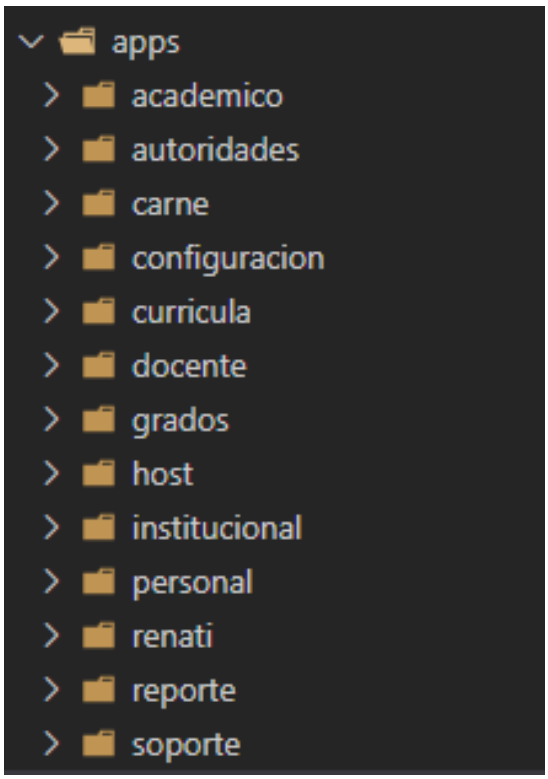



**Figura 39: Proyectos generados para los módulos del SIU**

**Nota: Elaboración propia**

Adicionalmente se creó un proyecto más, host, el cual será la aplicación contenedora y que cargará el resto de apps micro-frontend.

Se creó también una carpeta “libs” y dentro de ella se generó un proyecto de librería en el cual se separó el código de los componentes core para tener una mejor organización del proyecto.

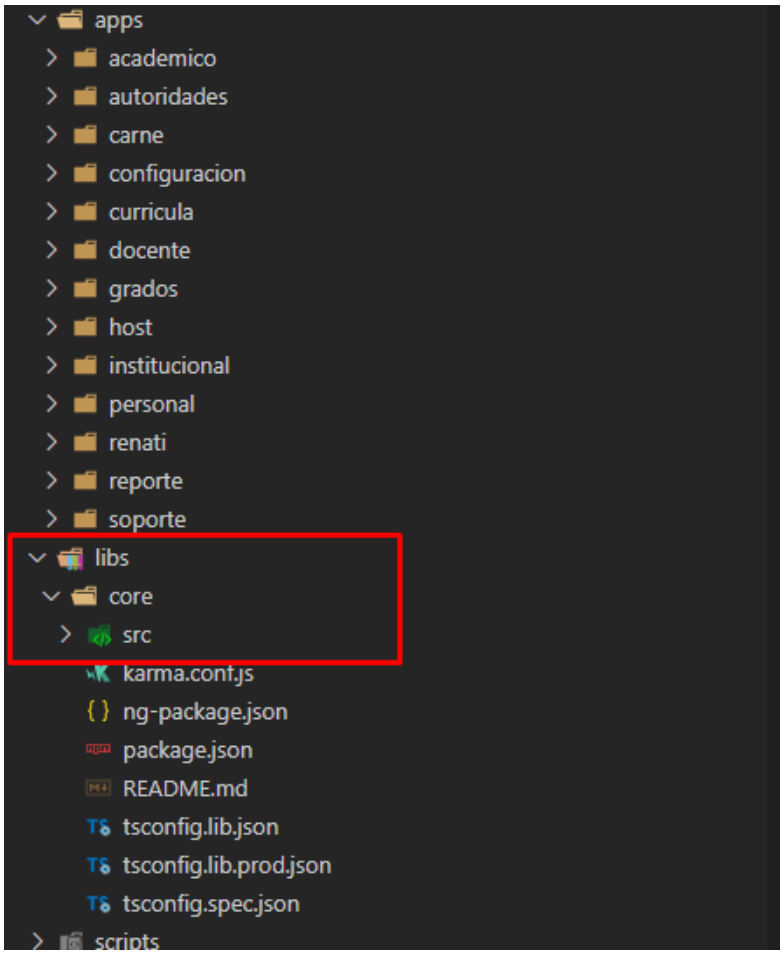



**Figura 40: Creación de carpeta libs y core para el código de los componentes core de la aplicación SIU**

**Nota: Elaboración propia**

## Implementación de Webpack Module Federation

Se ejecutó el comando para activar Module Federation en cada aplicación: "ng add @angular-architects/module-federation --project <nombre-app> --port <numero-puerto >"

```
>ng add @angular-architects/module-federation --project institucional --port 4201
```

**Figura 41: Ejemplo de comando para activar Angular Module federation en la app institucional**

**Nota: Elaboración propia**

El resultado del comando activó la característica Module Federation y creó por cada proyecto 2 archivos webpack.config.js (para los ambientes de desarrollo y producción), los cuales se presentan en la siguiente figura.

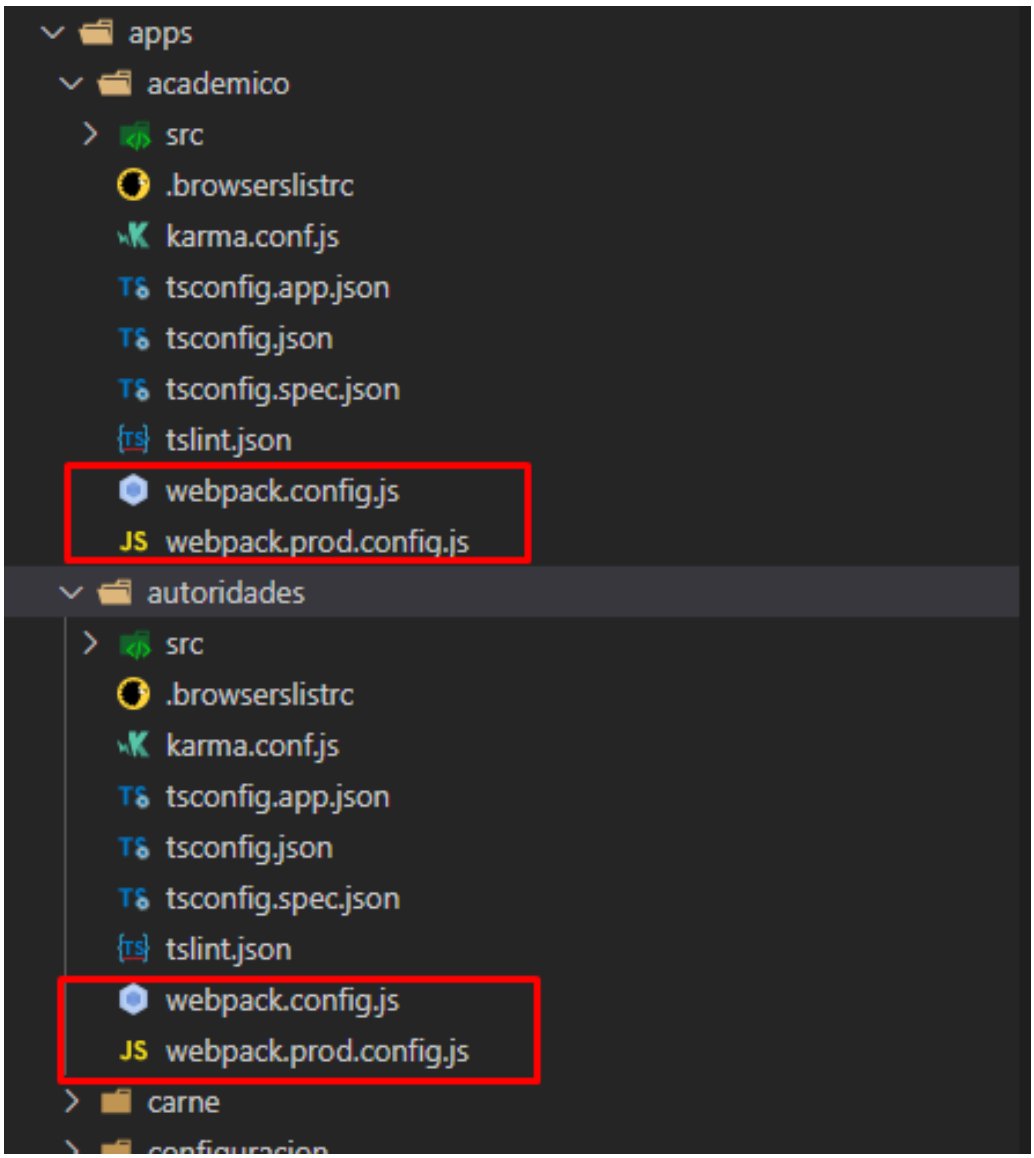



**Figura 42: Archivos generados al activar Module Federation en los proyectos**

**Nota: Elaboración propia**

Posteriormente se procedió a realizar en cada archivo webpack.config.js la configuración que permitirá exponer cada app de modo independiente bajo un propio site, se procedió a investigar cómo realizar la configuración que se adecúe más al proyecto y

que permita compartir entre las apps las librerías que son requeridas para su funcionamiento.

En la siguiente figura se presenta el contenido del archivo webpack.config.js luego de realizada la configuración que permite a las apps funcionar correctamente. Se resalta el nombre de la app, la cual variará según la app a la que pertenezca.

```
const sharedMappings = new mf.SharedMappings();
sharedMappings.register(
  path.join(__dirname, '../../tsconfig.json'),
  ['@siu/core']
);

module.exports = {
  output: {
    uniqueName: "configuracion",
    publicPath: "auto"
  },
  optimization: {
    runtimeChunk: false
  },
  resolve: {
    alias: {
      ...sharedMappings.getAliases(),
    }
  },
  plugins: [
    new ModuleFederationPlugin({

      name: "configuracion",
      filename: "remoteEntry.js",
      exposes: {
        './AppModule': `./src/app/app.module.ts`,
      },
      shared: {
        ...shareAll({
          singleton: true,
          strictVersion: true,
          requiredVersion: 'auto',
          eager: true
        }),
        ...share({
          "@angular/common/http": { singleton: true, strictVersion: true, eager: true },
          "@angular/cdk/overlay": { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          "@angular/material/input": { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/toolbar': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/icon': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/button': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/sidenav': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/list': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/menu': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/table': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/paginator': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/progress-bar': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/card': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/sort': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/select': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/form-field': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/checkbox': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/tooltip': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/dialog': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/snack-bar': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/expansion': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/datepicker': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/tabs': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/progress-spinner': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/radio': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/stepper': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/material/chips': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          '@angular/platform-browser/animations': { singleton: true, strictVersion: true, requiredVersion: 'auto', eager: true },
          ...sharedMappings.getDescriptors()
        }),
      }

    }),
    sharedMappings.getPlugin()
  ],
};
```

**Figura 43: Contenido del archivo webpack.config.js creado en cada proyecto**

**Nota: Elaboración propia**

Se procedió a implementar en este archivo la configuración adecuada para el proyecto web del SIU, especificando que paquetes se tendrá que compartir entre todas las apps. Este archivo exportará el contenido de cada app a través del archivo "remoteEntry.js", el cual serpa importada por la aplicación host, para que pueda cargar la app.

En el proyecto host se agregó el código presentado en la siguiente figura, el cual se encarga de definir cómo se cargarán las apps a través de rutas de la aplicación Angular mediante la obtención de un archivo json de configuración de apps llamado "mfe-config.json" el cual contiene la lista de apps del SIU y su ruta web, esto permitirá que la configuración de nuevas aplicaciones sea dinámica a través del archivo de configuración mfe-config.json.

```
5  import { Routes } from '@angular/router';
6  import { loadRemoteModule } from '@angular-architects/module-federation';
7  import { PLATFORM_ROUTES } from './app/platform-routes';
8  import { environment } from '@siu/host/environments/environment';
9
10 if (environment.production) {
11   enableProdMode();
12 }
13 fetch(environment.urls.mfeConfig, {
14   headers: {
15     'Cache-Control': 'no-cache'
16   }
17 }).then(async (res) => {
18   const config: { apps: any[] } = await res.json();
19
20   const platformRoutes: Routes = [];
21
22   config.apps.forEach(value => {
23     platformRoutes.push({
24       path: value.name,
25       loadChildren: () =>
26         loadRemoteModule({
27           remoteEntry: `${value.url}/remoteEntry.js?${new Date().getTime()}`,
28           remoteName: value.name,
29           exposedModule: './AppModule',
30         }).then((m) => m['AppModule']),
31     });
32   });
33
34   platformBrowserDynamic([
35     {
36       provide: PLATFORM_ROUTES,
37       useValue: platformRoutes,
38       multi: true,
39     },
40   ]).bootstrapModule(AppModule)
41     .catch(err => console.error(err));
42
43 });
```

**Figura 44: Código agregado en el proyecto host para definir las rutas Angular de las apps desde el archivo de configuración**

**Nota: Elaboración propia**

Se presenta también el formato del archivo de configuración mfe-config.json en la siguiente figura.

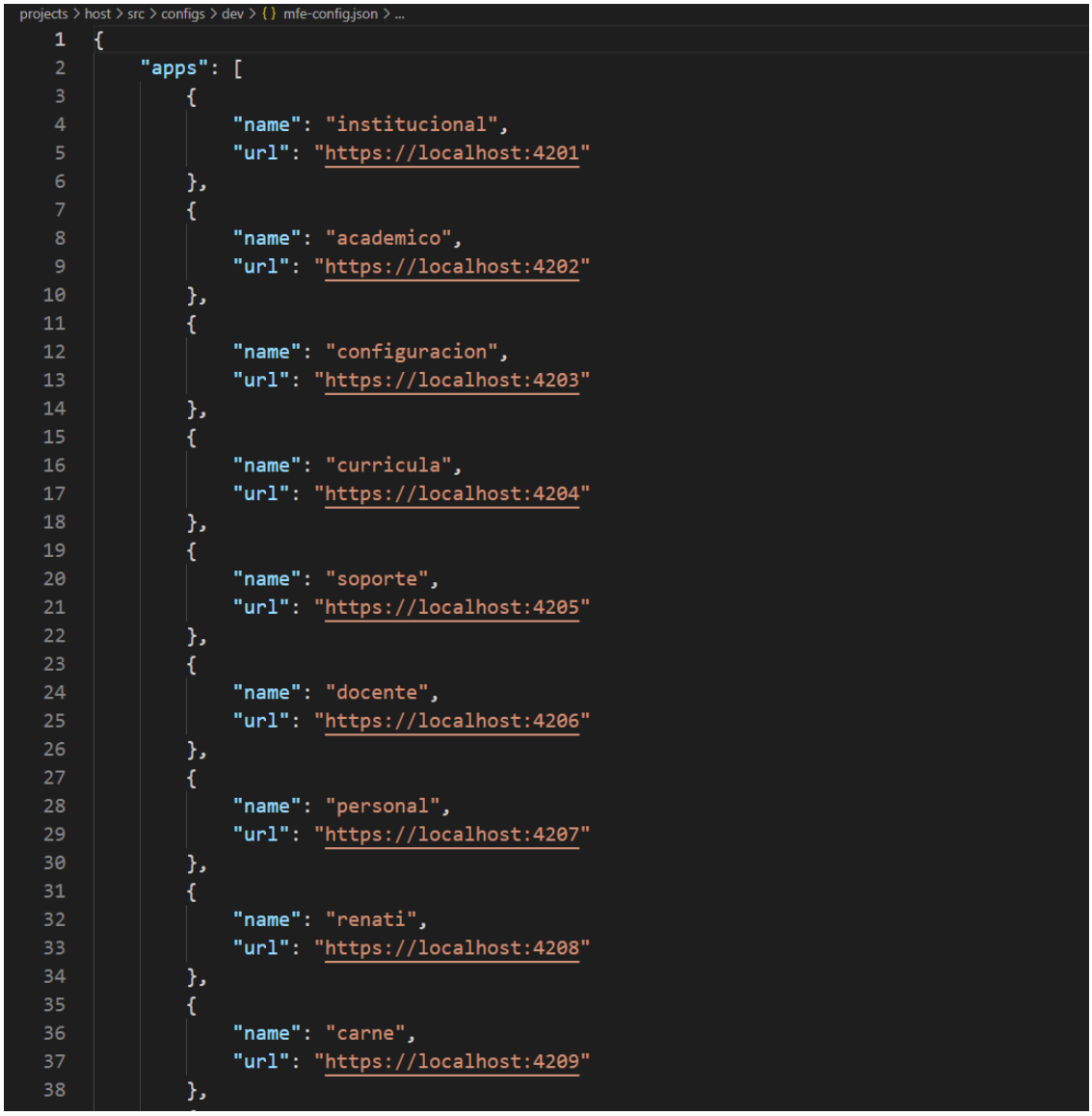



**Figura 45: Formato de definición de apps micro-frontend del SIU**

**Nota: Elaboración propia**

Luego de realizarse las configuraciones explicadas anteriormente se procedió a compilar los proyectos por separado y verificar que todo funcione correctamente.

A continuación, se presentarán los tiempos de compilación de la aplicación del SIU previo al desarrollo de la presente solución.

En la siguiente figura se observa que el tiempo de compilación es de 94649 ms o 1.57 minutos.

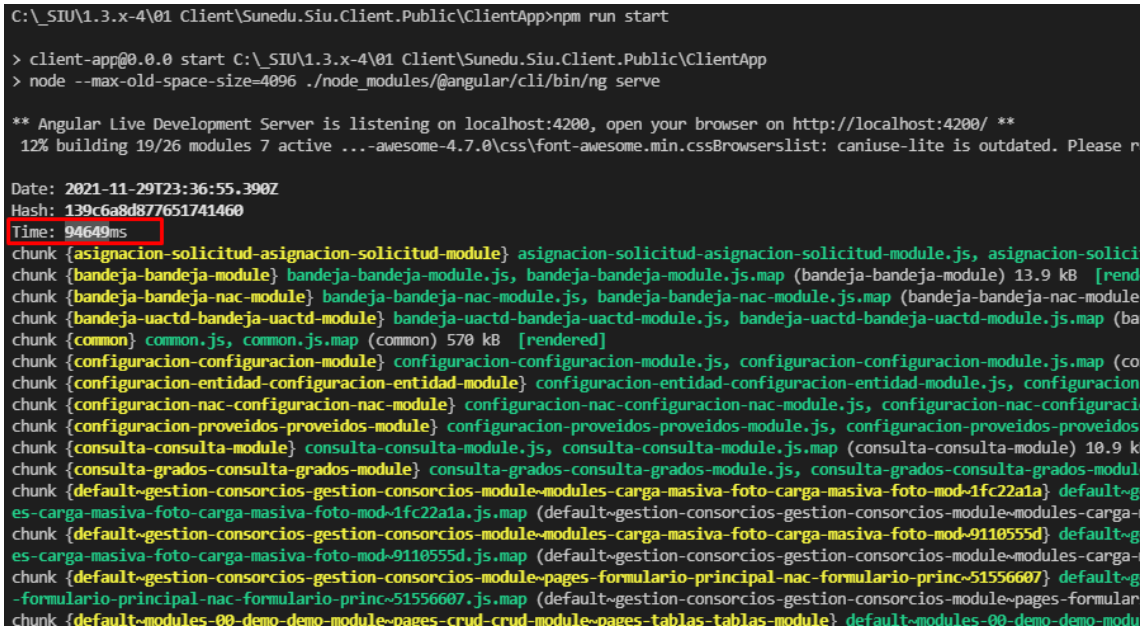



**Figura 46: Tiempo de la primera compilación de la aplicación web del SIU con la arquitectura monolítica**

**Nota: Elaboración propia**

En la siguiente figura se observa que el tiempo de compilación luego de un cambio en etapa de desarrollo es de 24534 ms o 24.5 segundos.


**Figura 47: Tiempo de compilación luego de un cambio en la etapa de desarrollo de la aplicación web del SIU con la arquitectura monolítica**

**Nota: Elaboración propia**

En la siguiente figura se observa que el tiempo de compilación para despliegue a producción es de 822014 ms o 13.7 minutos.

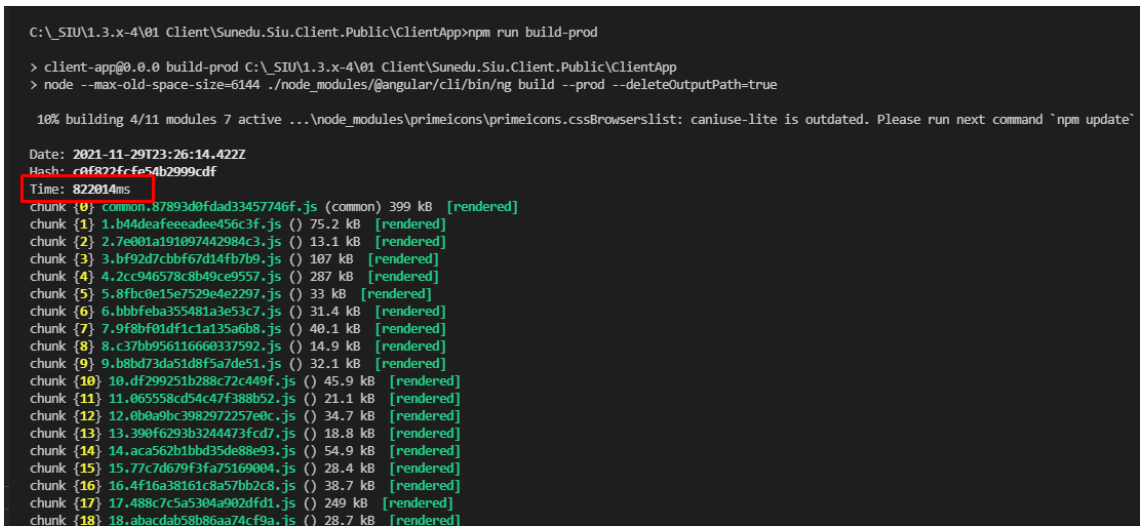



**Figura 48: Tiempo de compilación para desplegar en modo producción de la aplicación web del SIU con la arquitectura monolítica**

**Nota: Elaboración propia**

A continuación, se presentarán los tiempos de compilación de algunas apps de la aplicación del SIU con la implementación de la arquitectura micro-frontend.

En la siguiente figura se observa que el tiempo de compilación de la aplicación host es de 30011 ms o 30 segundos.

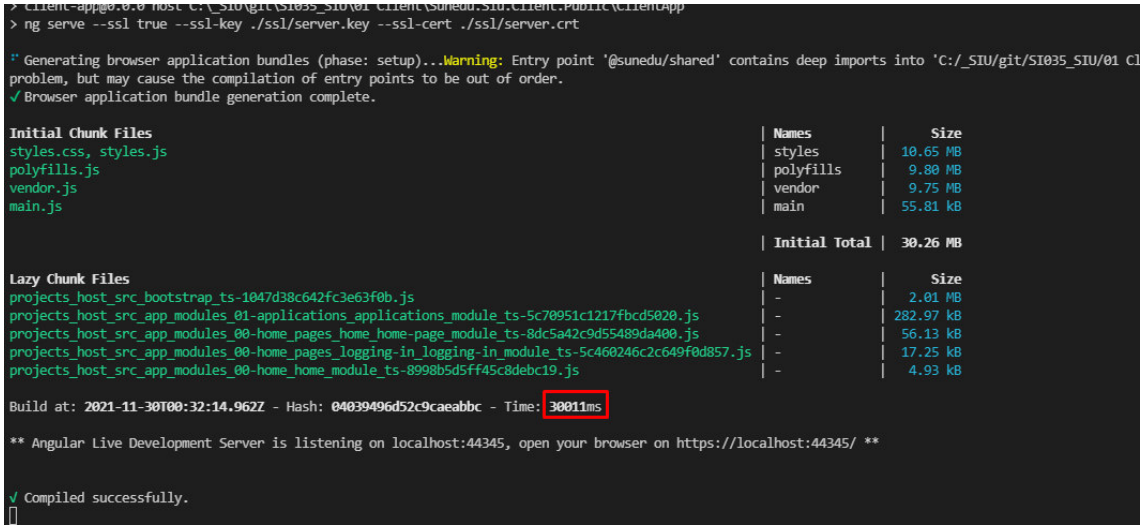



**Figura 49: Tiempo de compilación de la aplicación Host**

**Nota: Elaboración propia**

En la siguiente figura se observa que el tiempo de compilación de la aplicación grados es de 60361 ms o 60 segundos.

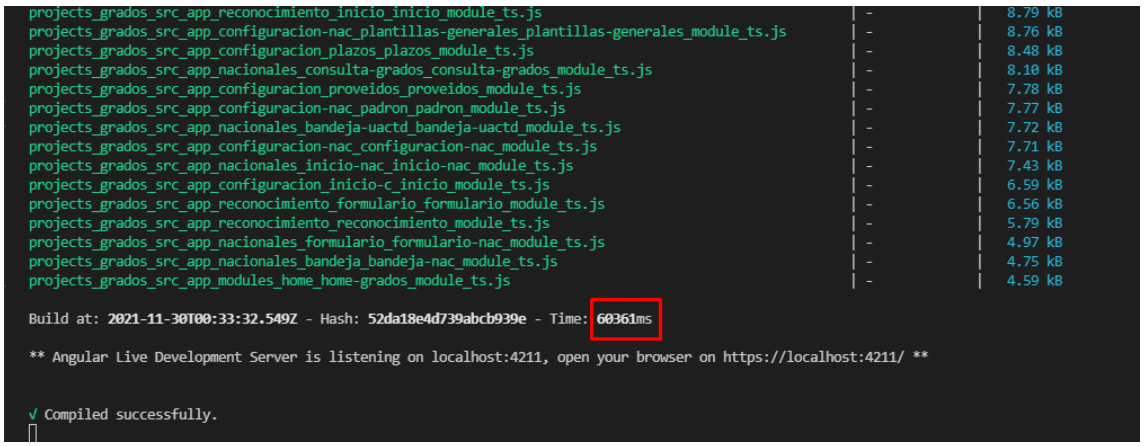



**Figura 50: Tiempo de compilación de la app de grados**

**Nota: Elaboración propia**

En la siguiente figura se observa que el tiempo de compilación de la aplicación grados es de 3337 ms o 3.3 segundos.

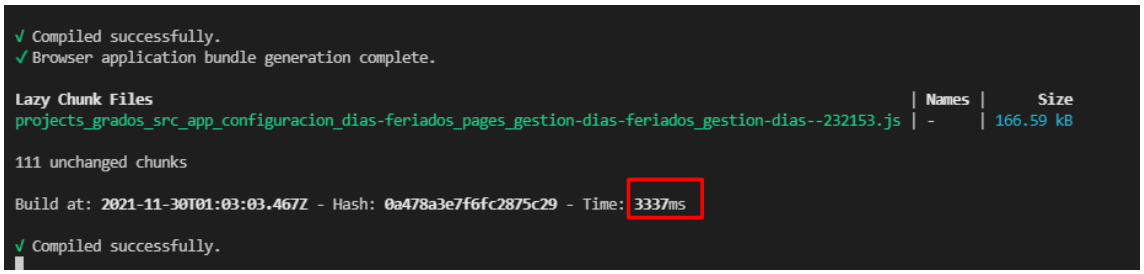



**Figura 51: Tiempo de compilación ante un cambio en desarrollo de la app de grados**

**Nota: Elaboración propia**

En la siguiente figura se observa que el tiempo de compilación para producción de la aplicación grados es de 78790 ms o 1.3 minutos.

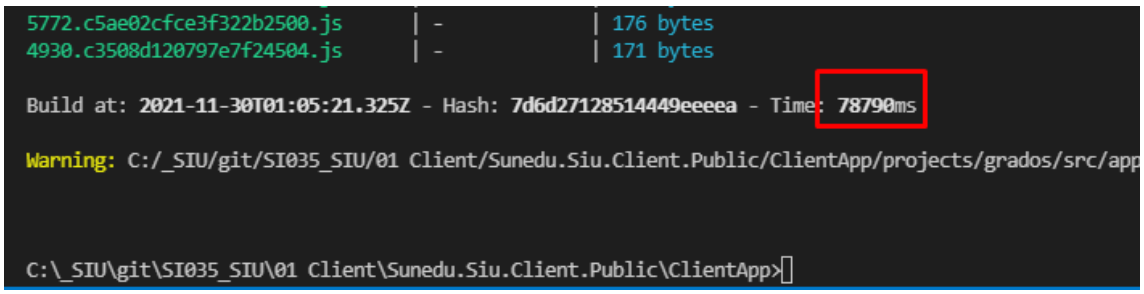



**Figura 52: Tiempo de compilación para desplegar en modo producción de la aplicación grados**

**Nota: Elaboración propia**

En la siguiente figura se observa que el tiempo de compilación de la aplicación institucional es de 46494 ms o 46.4 segundos.

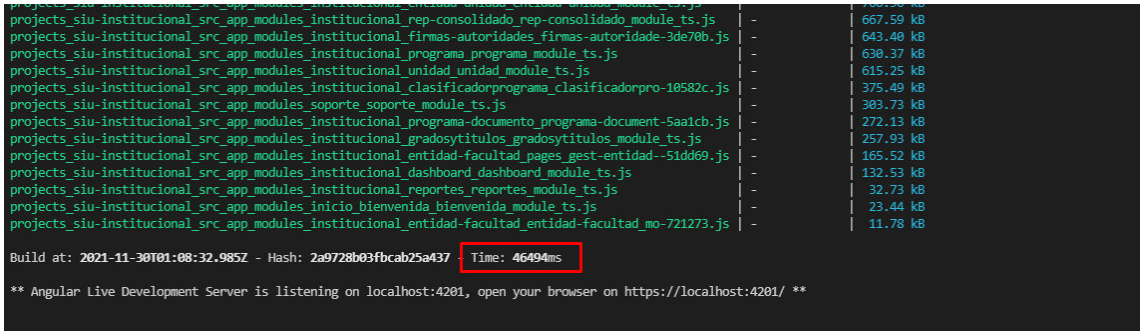



**Figura 53: Tiempo de compilación de la app institucional**

**Nota: Elaboración propia**

En la siguiente figura se observa que el tiempo de compilación de la aplicación institucional es de 1715 ms o 1.7 segundos.

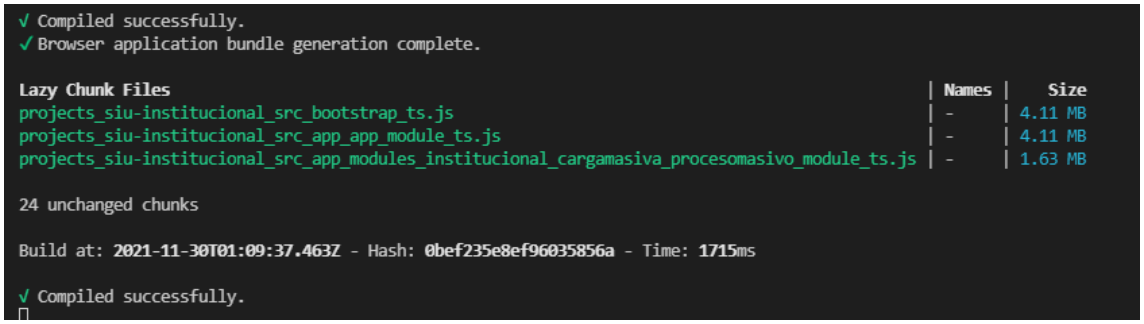



**Figura 54: Tiempo de compilación ante un cambio en desarrollo de la app institucional**

**Nota: Elaboración propia**

En la siguiente figura se observa que el tiempo de compilación para producción de la aplicación institucional es de 61436 ms o 1.02 minutos.

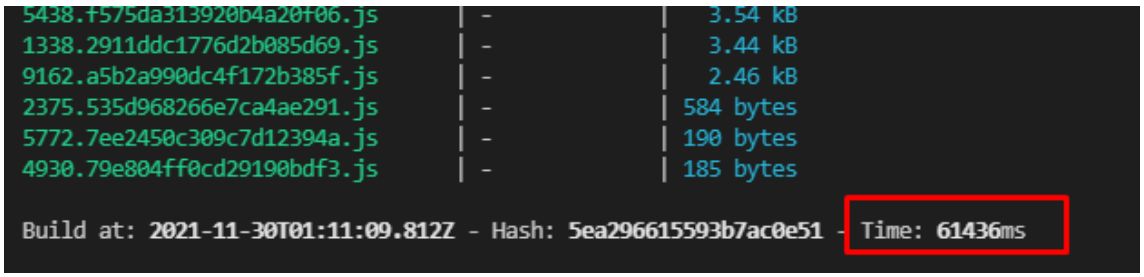



**Figura 55: Tiempo de compilación para desplegar en modo producción de la app institucional**

**Nota: Elaboración propia**

### 3.2.5.7. Sprint 6

#### Sprint Planning

Para empezar a obtener los beneficios de la nueva arquitectura de la aplicación web del SIU y luego de realizarse las pruebas correspondientes en el ambiente de desarrollo, fue necesario desplegar la aplicación en el ambiente de producción. Se planificó el despliegue de las apps al ambiente de producción.

#### Desarrollo del Sprint

El autor del presente informe participó parcialmente en el ítem N° 6 del Product Backlog: Despliegue en ambiente de desarrollo y pase a producción.

Se apoyó durante el despliegue de las apps que forman parte de la aplicación web del SIU en el ambiente de desarrollo y de producción (Se evidencia un correo de la reunión con el arquitecto en los anexos). Se creó el site SiuWeb en el servidor de producción para alojar a la app host (Ver figura 56) y para alojar a las apps micro-frontends se creó el site SiuWebModules (Ver figura 57) el cual las alojará a modo de aplicaciones del servidor. Las apps pudieron ser desplegadas cada una, de manera independiente, dentro de una aplicación creada dentro del site SiuWebModules. Este procedimiento se realizó de manera eficiente, cada app que se iba desplegando funcionaba correctamente sin afectar las apps que ya se encontraban desplegadas, de una manera más ordenada y mejorando así el procedimiento de despliegue que se realizaba anteriormente.

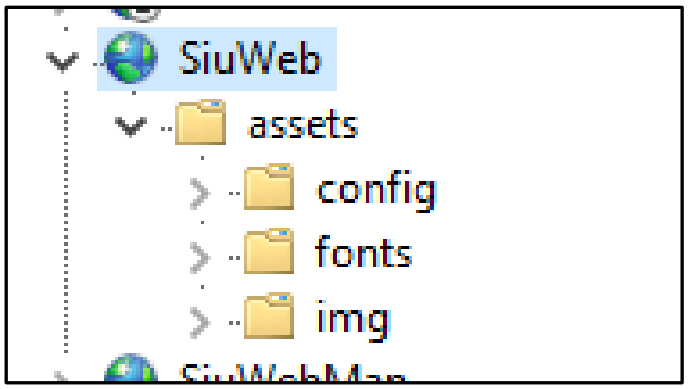



**Figura 56: Site creado en el servidor de producción para alojar a la app host**

**Nota: Elaboración propia**

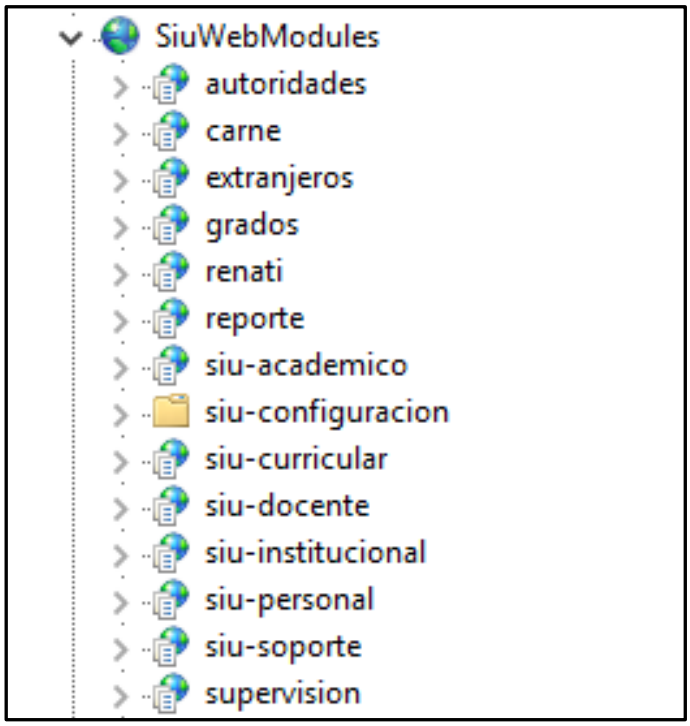



**Figura 57: Site creado en el servidor de producción para alojar a las apps micro-frontends del SIU**

**Nota: Elaboración propia**

Durante el periodo de 2 semanas a modo de marcha blanca se procedió a recabar información de los problemas hallados en el ambiente de producción, los cuales se fueron solucionando conforme se iban reportando. Se evidencia en los anexos, el correo por parte del arquitecto notificando el despliegue en los diversos ambientes incluido producción, así como el correo de la programación de una reunión de capacitación, impartida por el autor del informe, sobre la nueva arquitectura implementada hacia todo el equipo de desarrollo del SIU.

# CAPÍTULO IV
# REFLEXIÓN CRÍTICA DE LA EXPERIENCIA

La participación del autor del presente informe, quien se encargó de implementar esta arquitectura en la aplicación web del SIU, en este proyecto supuso un reto para el mismo, pues el concepto de micro-frontend es nuevo y no existe mucha documentación técnica al respecto, es por esto que había incertidumbre sobre la posibilidad de aplicar exitosamente el concepto en el SIU sin afectar el funcionamiento de la aplicación que ya se encontraba en producción.

Por otro lado, le sirvió al autor para adquirir nuevos conocimientos técnicos relativos a la implementación arquitecturas micro-frontend y cómo estos pueden optimizar el proceso de desarrollo e implementación de las aplicaciones web. Sirvió también para mejorar sus conocimientos sobre el framework Angular y de cómo ajustar este framework a las necesidades que un proyecto tiene.

El uso de la metodología Scrum en la implementación de este proyecto, utilizando los conceptos que esta propone, permitió dividir el proyecto en hitos cuyo resultado en cada uno de estos era el de un producto que ya podía ser distribuido a los equipos de desarrollo sin afectar el trabajo que ellos ya venían desarrollando, además permitió obtener los beneficios de la implementación de la arquitectura micro-frontend en corto tiempo.

Finalmente, se logró el objetivo de implementar un concepto de arquitectura frontend nuevo, pero que permita mejorar el proceso de desarrollo, implementación y despliegue de la aplicación web del SIU, y que obtuvo comentarios positivos de los equipos de desarrollo del sistema SIU.

# CAPÍTULO V

# CONCLUSIONES Y RECOMENDACIONES

## 5.1 CONCLUSIONES

Se logró implementar la arquitectura micro-frontend en la aplicación web frontend del SIU mediante la tecnología webpack module federation, la cual funciona adecuadamente y una vez solucionados los errores reportados durante la marcha blanca, se concluye que esta implementación no afectó las funcionalidades que ya se encontraban desarrolladas.

Se dividió la aplicación web frontend del SIU en pequeñas apps, logrando disminuir los tiempos de compilación y despliegue de la aplicación web tal y como se observa desde la figura 49 hasta la figura 55, lo cual supone una mejora importante en la reducción de tiempos en las fases de desarrollo y despliegue de la aplicación web del SIU.

Se crearon sitios web en el servidor que permiten alojar cada app micro-frontend del SIU, lo cual optimizó el procedimiento de despliegue de la aplicación web del SIU, logrando un despliegue independiente de app, esto sin que afecte las apps que ya están desplegadas. Al estar alojadas en un sitio web distinto, el despliegue de una app, no supone un riesgo para las apps que ya se encuentran en funcionamiento.

El uso de la metodología Scrum permitió obtener los resultados esperados y valor para todo el equipo de desarrollo del SIU en un corto plazo. Las reuniones y retroalimentación fueron fundamentales para la correcta implementación de este proyecto.

## 5.2 RECOMENDACIONES

De la implementación de la arquitectura micro-frontend en el sistema de información universitaria SIU, el autor del presente informe recomienda la evaluación de las siguientes propuestas:

- Si bien las apps implementadas en la arquitectura son independientes, están desplegadas en un mismo servidor web de la SUNEDU, se recomienda dar el siguiente paso hacia un despliegue en la nube usando contenedores para cada una de las apps.
- Debido al impacto que obtuvo este proyecto en cuanto a optimización de tiempos y recursos se recomienda aplicar esta arquitectura en los otros sistemas de gran envergadura a cargo de la SUNEDU.

## 5.3 FUENTES DE INFORMACIÓN


Ali, J. (2013). *Instant Node Package Manager.* Packt Publishing Ltd.

Angular. (2021). *Angular - Whats is Angular?* Obtenido de https://angular.io/: https://angular.io/guide/what-is-angular

Angular. (2021). *Angular Update Guide*. Obtenido de https://update.angular.io/

Deemer, P., Benefield, G., Larman, C., & Vodde, B. (2009). *THE SCRUM PRIMER.* California: Scrum Training Institute.

Doglio, F. (2015). *Pro REST API Development with Node.js.* Apress.

Eslava Muñoz, V. J. (2012). *HTML, presente y futuro de la web.*

Git. (2021). *Git*. Obtenido de https://git-scm.com/

Jadhav, M. A., Sawant, B. R., & Deshmukh, A. (2015). Single page application using angularjs. *International Journal of Computer Science and Information Technologies, 6*(3), 2876-2879.

López, D., & Maya, E. (2017). *Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web.*

Luna Del Águila, F. (1999). *Implementación del Protocolo HTTP Paralelizado en Cliente y Servidor.*

Mezzalira, L. (2021). *Building Micro-Frontends.*

*Node.js*. (2021). Obtenido de https://nodejs.org/es/

NPM. (2021). *NPM*. Obtenido de https://docs.npmjs.com: https://docs.npmjs.com/about-npm

Nurseitov, N., Paulson, M., Reynolds, R., & Izurieta, C. (2009). Comparison of JSON and XML data interchange formats: a case study. *Caine, 9*, 157-162.

Pérez, J. E. (2009). *Introduccion A Javascript.*

*Sunedu*. (2021). Obtenido de https://www.sunedu.gob.pe/wp-content/uploads/2018/04/Informe-de-Transferencia-2018.pdf

*Sunedu*. (2021). Obtenido de https://www.sunedu.gob.pe/: https://www.sunedu.gob.pe/organigrama-2/

Trigas Gallego, M. (2012). *Metodologia scrum.*

Typescript. (2021). *Typescript*. Obtenido de https://www.typescriptlang.org/

Webpack. (2021). *Webpack*. Obtenido de Webpack: https://github.com/webpack/webpack

*www.gob.pe*. (2021). Obtenido de https://www.gob.pe/4504-superintendencia-nacional-de-educacion-superior-universitaria-que-hacemos

## 5.4 GLOSARIO

**SUNEDU:** Superintendencia Nacional de Educación Superior Universitaria.

**SIU:** Sistema de información universitaria.

**Frontend:** Es la capa visual de un sistema informático.

**Backend:** Es la capa invisible de un proyecto web que se encarga de los procesos necesarios para que el proyecto funcione correctamente.

**Software:** Soporte lógico que hace posible la realización de las tareas de un sistema informático.

**Open Source:** Código o software accesible al público.

**Framework:** Entorno de trabajo que se utiliza como estructura base para el desarrollo de un proyecto de software.

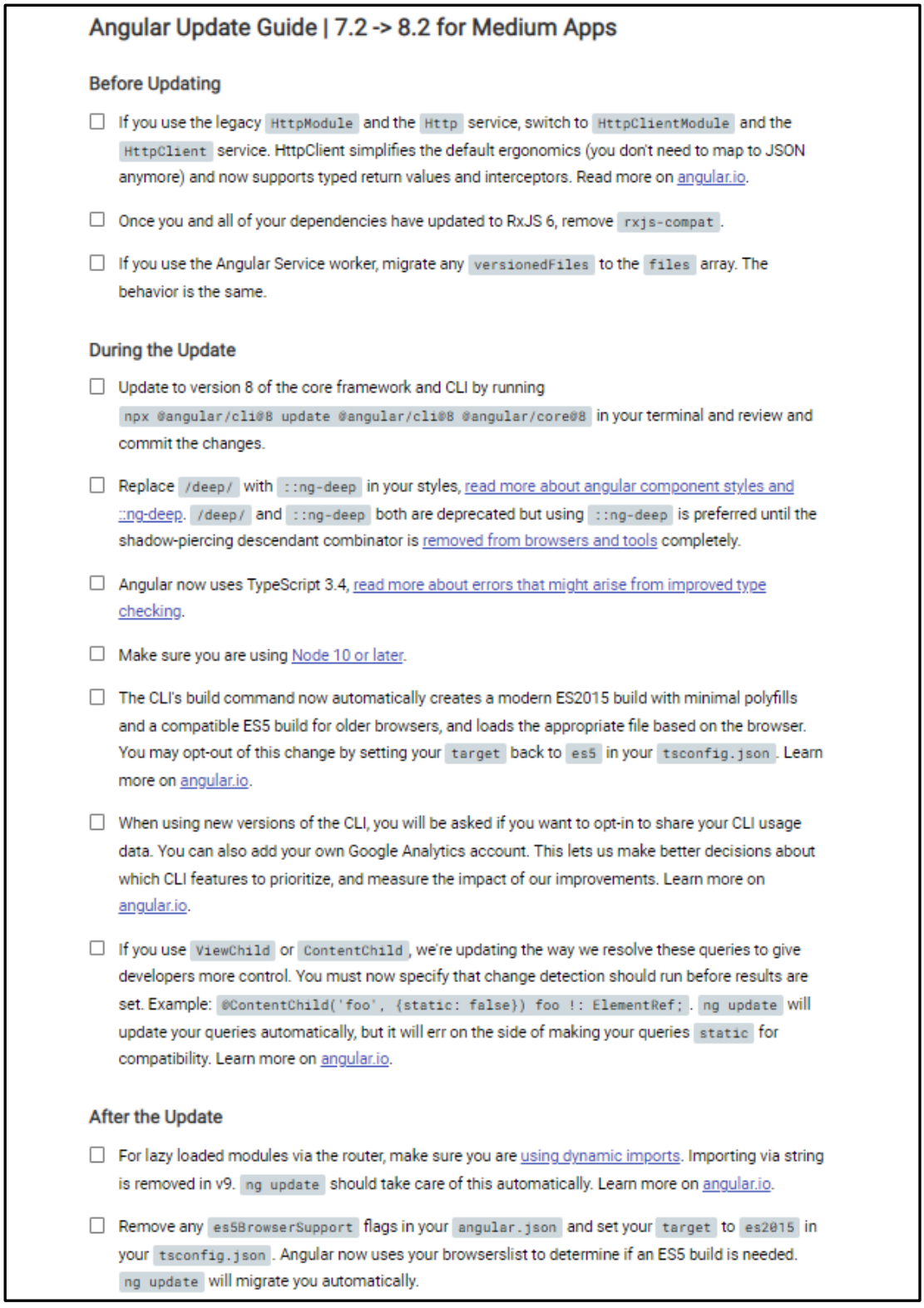
**Arquitectura monolítica:** Es un modelo de construcción de software en el que todos los aspectos funcionales se encuentran acoplados dentro de un mismo programa.

**Interface:** Conexión funcional entre 2 sistemas.

**Site**: Servidor de internet, que aloja páginas o aplicaciones web.

# ANEXOS

## ANEXO 1. Guías de migración de Angular Framework



### Angular Update Guide | 7.2 -> 8.2 for Medium Apps

#### Before Updating

- [ ] If you use the legacy `HttpModule` and the `Http` service, switch to `HttpClientModule` and the `HttpClient` service. HttpClient simplifies the default ergonomics (you don't need to map to JSON anymore) and now supports typed return values and interceptors. Read more on angular.io.
- [ ] Once you and all of your dependencies have updated to RxJS 6, remove `rxjs-compat`.
- [ ] If you use the Angular Service worker, migrate any `versionedFiles` to the `files` array. The behavior is the same.

#### During the Update

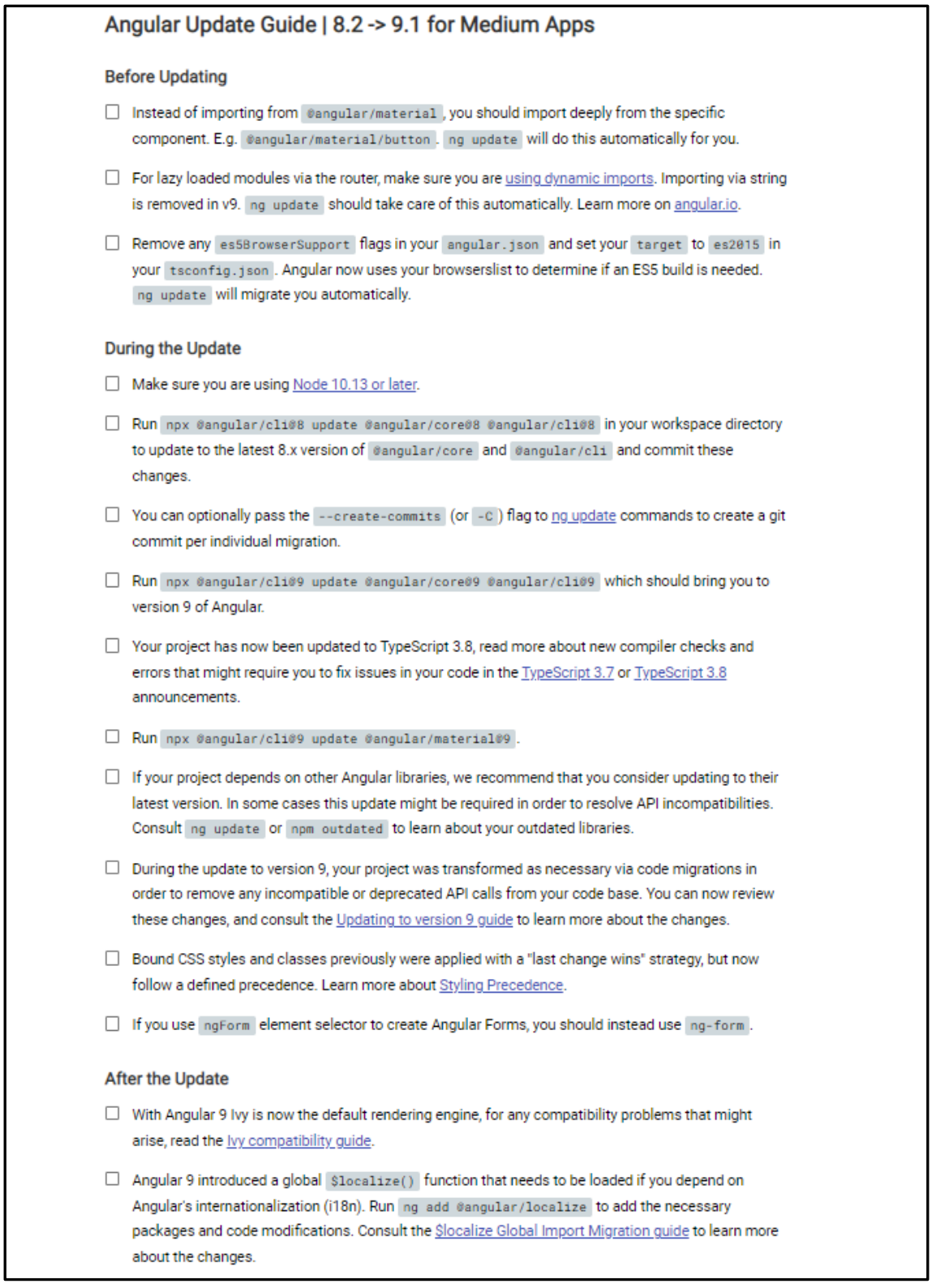
- [ ] Update to version 8 of the core framework and CLI by running `npx @angular/cli@8 update @angular/cli@8 @angular/core@8` in your terminal and review and commit the changes.
- [ ] Replace `/deep/` with `::ng-deep` in your styles, read more about angular component styles and ::ng-deep. `/deep/` and `::ng-deep` both are deprecated but using `::ng-deep` is preferred until the shadow-piercing descendant combinator is removed from browsers and tools completely.
- [ ] Angular now uses TypeScript 3.4, read more about errors that might arise from improved type checking.
- [ ] Make sure you are using Node 10 or later.
- [ ] The CLI's build command now automatically creates a modern ES2015 build with minimal polyfills and a compatible ES5 build for older browsers, and loads the appropriate file based on the browser. You may opt-out of this change by setting your `target` back to `es5` in your `tsconfig.json`. Learn more on angular.io.
- [ ] When using new versions of the CLI, you will be asked if you want to opt-in to share your CLI usage data. You can also add your own Google Analytics account. This lets us make better decisions about which CLI features to prioritize, and measure the impact of our improvements. Learn more on angular.io.
- [ ] If you use `ViewChild` or `ContentChild`, we're updating the way we resolve these queries to give developers more control. You must now specify that change detection should run before results are set. Example: `@ContentChild('foo', {static: false}) foo !: ElementRef;`. `ng update` will update your queries automatically, but it will err on the side of making your queries `static` for compatibility. Learn more on angular.io.

#### After the Update

- [ ] For lazy loaded modules via the router, make sure you are using dynamic imports. Importing via string is removed in v9. `ng update` should take care of this automatically. Learn more on angular.io.
- [ ] Remove any `es5BrowserSupport` flags in your `angular.json` and set your `target` to `es2015` in your `tsconfig.json`. Angular now uses your browserslist to determine if an ES5 build is needed. `ng update` will migrate you automatically.

**Figura 58: Guía de migración de Angular versión 7.2 a 8.2**

**Nota: Adaptado de Angular Update Guide (2021)**



## Angular Update Guide | 8.2 -> 9.1 for Medium Apps

### Before Updating

- [ ] Instead of importing from `@angular/material`, you should import deeply from the specific component. E.g. `@angular/material/button`. `ng update` will do this automatically for you.
- [ ] For lazy loaded modules via the router, make sure you are using dynamic imports. Importing via string is removed in v9. `ng update` should take care of this automatically. Learn more on angular.io.
- [ ] Remove any `es5BrowserSupport` flags in your `angular.json` and set your `target` to `es2015` in your `tsconfig.json`. Angular now uses your browserslist to determine if an ES5 build is needed. `ng update` will migrate you automatically.

### During the Update

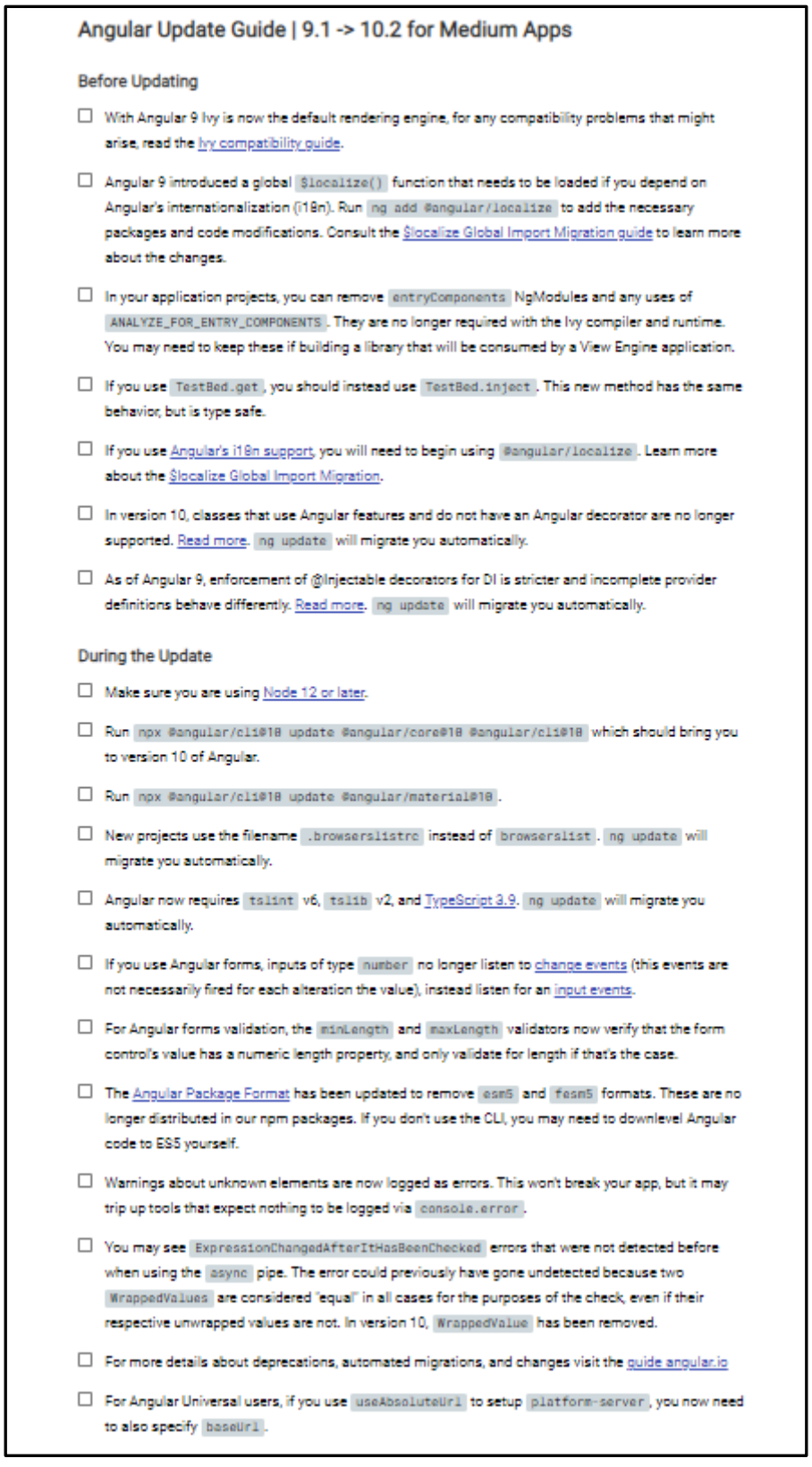
- [ ] Make sure you are using Node 10.13 or later.
- [ ] Run `npx @angular/cli@8 update @angular/core@8 @angular/cli@8` in your workspace directory to update to the latest 8.x version of `@angular/core` and `@angular/cli` and commit these changes.
- [ ] You can optionally pass the `--create-commits` (or `-C`) flag to ng update commands to create a git commit per individual migration.
- [ ] Run `npx @angular/cli@9 update @angular/core@9 @angular/cli@9` which should bring you to version 9 of Angular.
- [ ] Your project has now been updated to TypeScript 3.8, read more about new compiler checks and errors that might require you to fix issues in your code in the TypeScript 3.7 or TypeScript 3.8 announcements.
- [ ] Run `npx @angular/cli@9 update @angular/material@9`.
- [ ] If your project depends on other Angular libraries, we recommend that you consider updating to their latest version. In some cases this update might be required in order to resolve API incompatibilities. Consult `ng update` or `npm outdated` to learn about your outdated libraries.
- [ ] During the update to version 9, your project was transformed as necessary via code migrations in order to remove any incompatible or deprecated API calls from your code base. You can now review these changes, and consult the Updating to version 9 guide to learn more about the changes.
- [ ] Bound CSS styles and classes previously were applied with a "last change wins" strategy, but now follow a defined precedence. Learn more about Styling Precedence.
- [ ] If you use `ngForm` element selector to create Angular Forms, you should instead use `ng-form`.

### After the Update

- [ ] With Angular 9 Ivy is now the default rendering engine, for any compatibility problems that might arise, read the Ivy compatibility guide.
- [ ] Angular 9 introduced a global `$localize()` function that needs to be loaded if you depend on Angular's internationalization (i18n). Run `ng add @angular/localize` to add the necessary packages and code modifications. Consult the $localize Global Import Migration guide to learn more about the changes.

**Figura 59: Guía de migración de Angular versión 8.2 a 9.1**

**Nota: Adaptado de Angular Update Guide (2021)**



## Angular Update Guide | 9.1 -> 10.2 for Medium Apps

### Before Updating

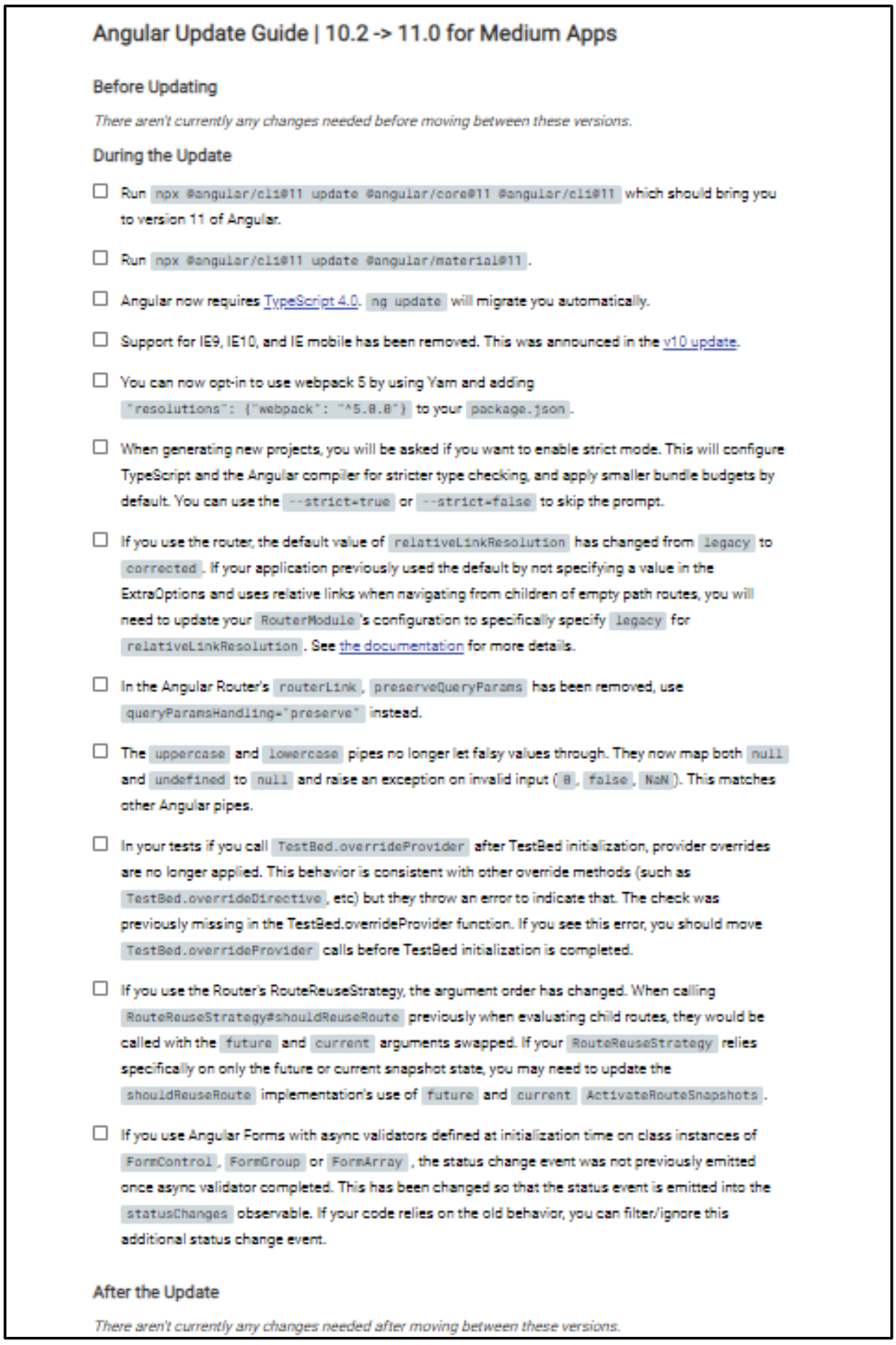
- ☐ With Angular 9 Ivy is now the default rendering engine, for any compatibility problems that might arise, read the Ivy compatibility guide.
- ☐ Angular 9 introduced a global `$localize()` function that needs to be loaded if you depend on Angular's internationalization (i18n). Run `ng add @angular/localize` to add the necessary packages and code modifications. Consult the $localize Global Import Migration guide to learn more about the changes.
- ☐ In your application projects, you can remove `entryComponents` NgModules and any uses of `ANALYZE_FOR_ENTRY_COMPONENTS`. They are no longer required with the Ivy compiler and runtime. You may need to keep these if building a library that will be consumed by a View Engine application.
- ☐ If you use `TestBed.get`, you should instead use `TestBed.inject`. This new method has the same behavior, but is type safe.
- ☐ If you use Angular's i18n support, you will need to begin using `@angular/localize`. Learn more about the $localize Global Import Migration.
- ☐ In version 10, classes that use Angular features and do not have an Angular decorator are no longer supported. Read more. `ng update` will migrate you automatically.
- ☐ As of Angular 9, enforcement of @Injectable decorators for DI is stricter and incomplete provider definitions behave differently. Read more. `ng update` will migrate you automatically.

### During the Update

- ☐ Make sure you are using Node 12 or later.
- ☐ Run `npx @angular/cli@10 update @angular/core@10 @angular/cli@10` which should bring you to version 10 of Angular.
- ☐ Run `npx @angular/cli@10 update @angular/material@10`.
- ☐ New projects use the filename `.browserslistrc` instead of `browserslist`. `ng update` will migrate you automatically.
- ☐ Angular now requires `tslint` v6, `tslib` v2, and TypeScript 3.9. `ng update` will migrate you automatically.
- ☐ If you use Angular forms, inputs of type `number` no longer listen to change events (this events are not necessarily fired for each alteration the value), instead listen for an input events.
- ☐ For Angular forms validation, the `minLength` and `maxLength` validators now verify that the form control's value has a numeric length property, and only validate for length if that's the case.
- ☐ The Angular Package Format has been updated to remove `esm5` and `fesm5` formats. These are no longer distributed in our npm packages. If you don't use the CLI, you may need to downlevel Angular code to ES5 yourself.
- ☐ Warnings about unknown elements are now logged as errors. This won't break your app, but it may trip up tools that expect nothing to be logged via `console.error`.
- ☐ You may see `ExpressionChangedAfterItHasBeenChecked` errors that were not detected before when using the `async` pipe. The error could previously have gone undetected because two `WrappedValues` are considered "equal" in all cases for the purposes of the check, even if their respective unwrapped values are not. In version 10, `WrappedValue` has been removed.
- ☐ For more details about deprecations, automated migrations, and changes visit the guide.angular.io
- ☐ For Angular Universal users, if you use `useAbsoluteUrl` to setup `platform-server`, you now need to also specify `baseUrl`.

**Figura 60: Guía de migración de Angular versión 9.1 a 10.2**

**Nota: Adaptado de Angular Update Guide (2021)**



## Angular Update Guide | 10.2 -> 11.0 for Medium Apps

### Before Updating

*There aren't currently any changes needed before moving between these versions.*

### During the Update

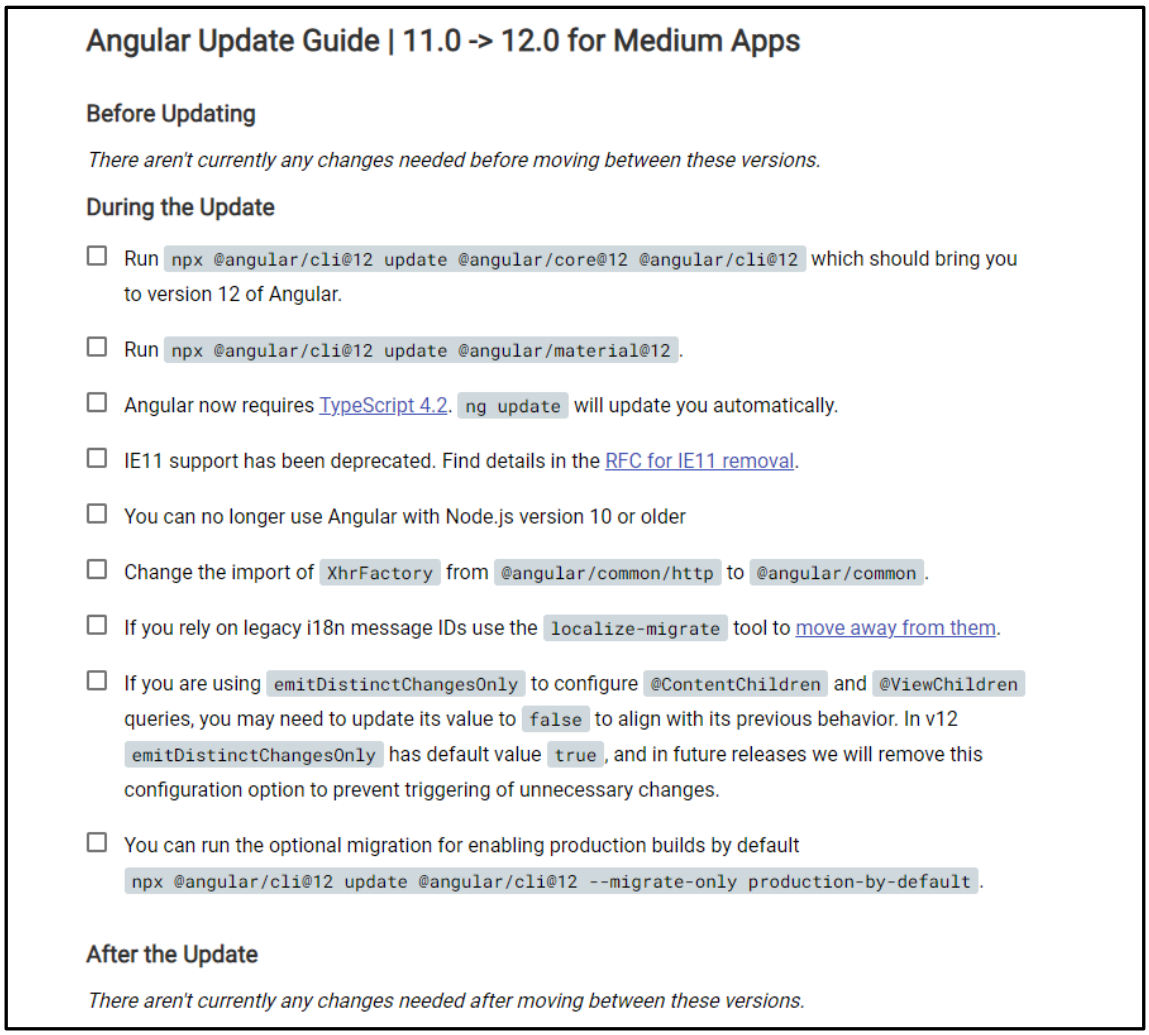
- ☐ Run `npx @angular/cli@11 update @angular/core@11 @angular/cli@11` which should bring you to version 11 of Angular.
- ☐ Run `npx @angular/cli@11 update @angular/material@11`.
- ☐ Angular now requires TypeScript 4.0. `ng update` will migrate you automatically.
- ☐ Support for IE9, IE10, and IE mobile has been removed. This was announced in the v10 update.
- ☐ You can now opt-in to use webpack 5 by using Yarn and adding `"resolutions": {"webpack": "^5.0.0"}` to your `package.json`.
- ☐ When generating new projects, you will be asked if you want to enable strict mode. This will configure TypeScript and the Angular compiler for stricter type checking, and apply smaller bundle budgets by default. You can use the `--strict=true` or `--strict=false` to skip the prompt.
- ☐ If you use the router, the default value of `relativeLinkResolution` has changed from `legacy` to `corrected`. If your application previously used the default by not specifying a value in the ExtraOptions and uses relative links when navigating from children of empty path routes, you will need to update your `RouterModule`'s configuration to specifically specify `legacy` for `relativeLinkResolution`. See the documentation for more details.
- ☐ In the Angular Router's `routerLink`, `preserveQueryParams` has been removed, use `queryParamsHandling="preserve"` instead.
- ☐ The `uppercase` and `lowercase` pipes no longer let falsy values through. They now map both `null` and `undefined` to `null` and raise an exception on invalid input (`0`, `false`, `NaN`). This matches other Angular pipes.
- ☐ In your tests if you call `TestBed.overrideProvider` after TestBed initialization, provider overrides are no longer applied. This behavior is consistent with other override methods (such as `TestBed.overrideDirective`, etc) but they throw an error to indicate that. The check was previously missing in the TestBed.overrideProvider function. If you see this error, you should move `TestBed.overrideProvider` calls before TestBed initialization is completed.
- ☐ If you use the Router's RouteReuseStrategy, the argument order has changed. When calling `RouteReuseStrategy#shouldReuseRoute` previously when evaluating child routes, they would be called with the `future` and `current` arguments swapped. If your `RouteReuseStrategy` relies specifically on only the future or current snapshot state, you may need to update the `shouldReuseRoute` implementation's use of `future` and `current` `ActivateRouteSnapshots`.
- ☐ If you use Angular Forms with async validators defined at initialization time on class instances of `FormControl`, `FormGroup` or `FormArray`, the status change event was not previously emitted once async validator completed. This has been changed so that the status event is emitted into the `statusChanges` observable. If your code relies on the old behavior, you can filter/ignore this additional status change event.

### After the Update

*There aren't currently any changes needed after moving between these versions.*

**Figura 61: Guía de migración de Angular versión 10.2 a 11.0**

**Nota: Adaptado de Angular Update Guide (2021)**



## Angular Update Guide | 11.0 -> 12.0 for Medium Apps

### Before Updating

*There aren't currently any changes needed before moving between these versions.*

### During the Update

- [ ] Run `npx @angular/cli@12 update @angular/core@12 @angular/cli@12` which should bring you to version 12 of Angular.
- [ ] Run `npx @angular/cli@12 update @angular/material@12`.
- [ ] Angular now requires TypeScript 4.2. `ng update` will update you automatically.
- [ ] IE11 support has been deprecated. Find details in the RFC for IE11 removal.
- [ ] You can no longer use Angular with Node.js version 10 or older
- [ ] Change the import of `XhrFactory` from `@angular/common/http` to `@angular/common`.
- [ ] If you rely on legacy i18n message IDs use the `localize-migrate` tool to move away from them.
- [ ] If you are using `emitDistinctChangesOnly` to configure `@ContentChildren` and `@ViewChildren` queries, you may need to update its value to `false` to align with its previous behavior. In v12 `emitDistinctChangesOnly` has default value `true`, and in future releases we will remove this configuration option to prevent triggering of unnecessary changes.
- [ ] You can run the optional migration for enabling production builds by default `npx @angular/cli@12 update @angular/cli@12 --migrate-only production-by-default`.

### After the Update

*There aren't currently any changes needed after moving between these versions.*

**Figura 62: Guía de migración de Angular versión 11.0 a 12.0**

**Nota: Adaptado de Angular Update Guide (2021)**

## ANEXO 2. Pantallas de la aplicación SIU

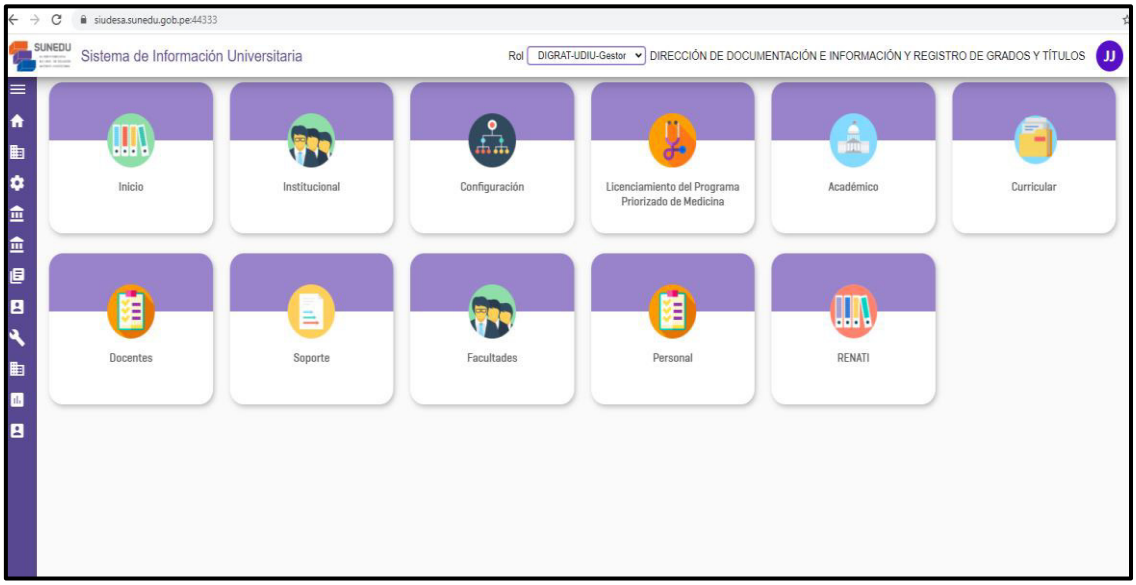



**Figura 63: Pantalla con la página principal del SIU**

**Nota: Elaboración propia**


**Figura 64: Pantalla con las opciones de la app institucional**

**Nota: Elaboración propia**

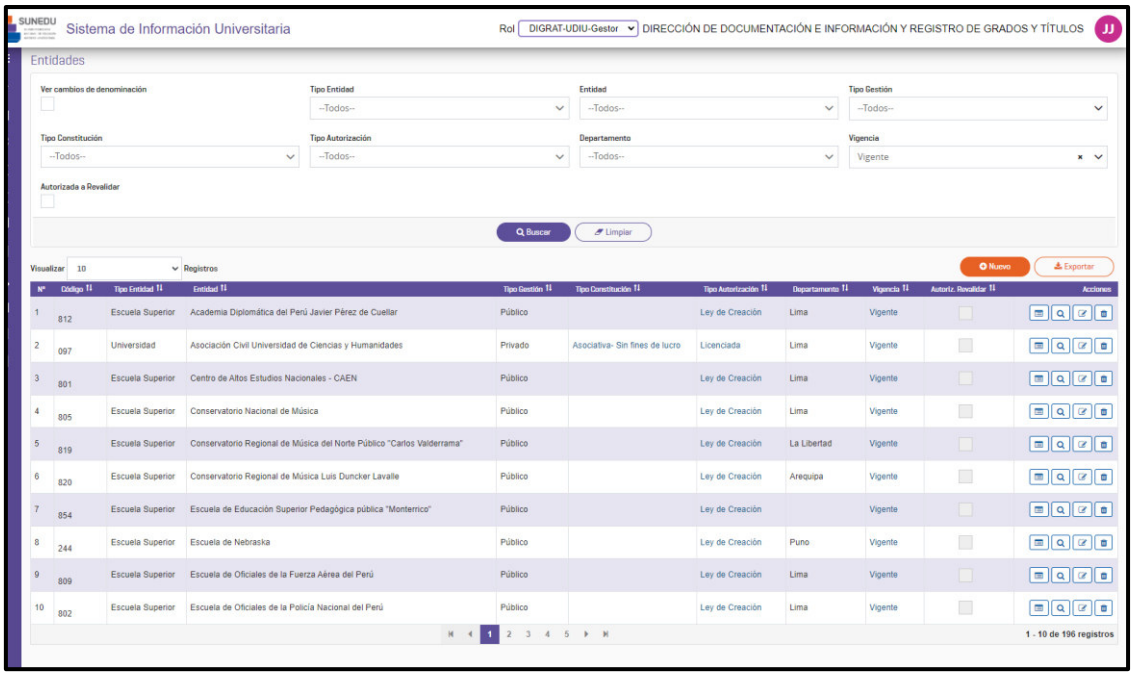



**Figura 65: Pantalla de la funcionalidad de gestión de entidades**

**Nota: Elaboración propia**

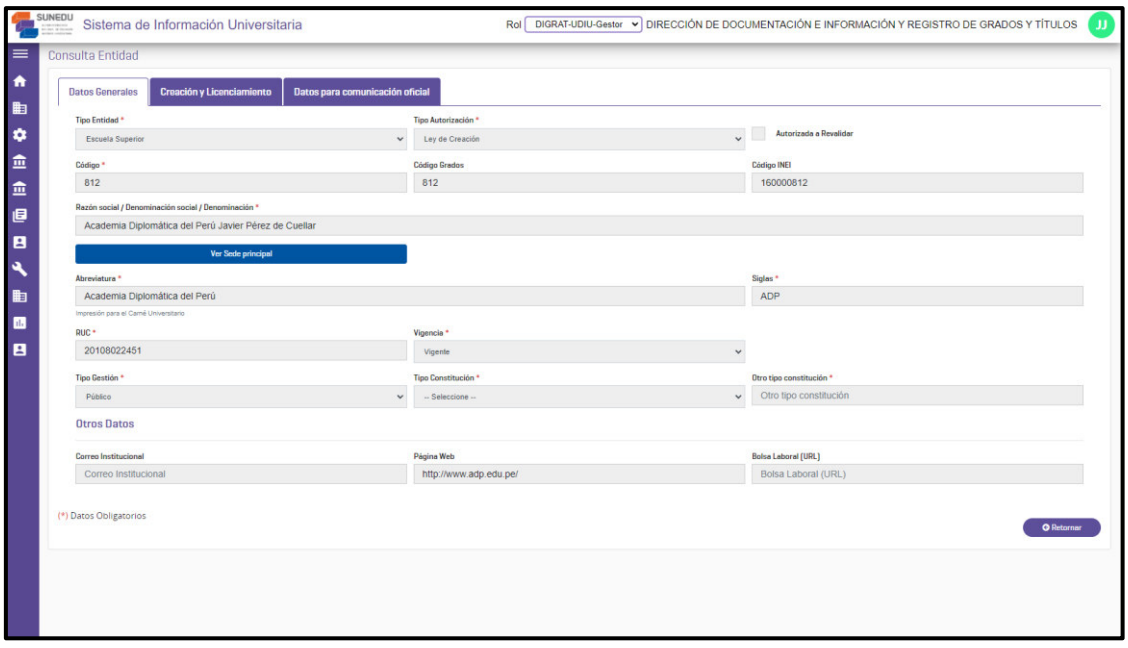



**Figura 66: Pantalla de la funcionalidad de consulta de entidad**

**Nota: Elaboración propia**

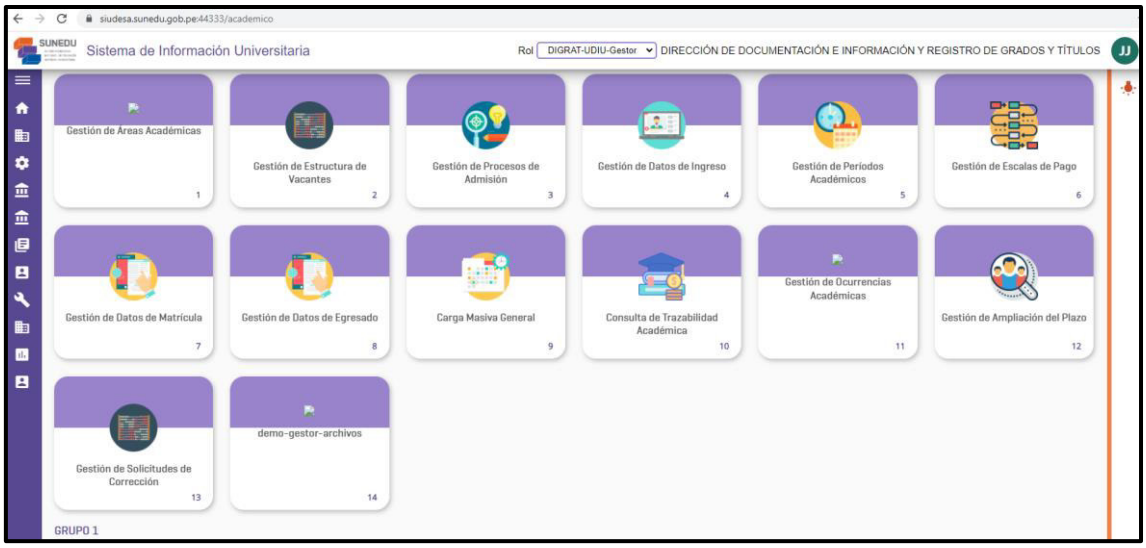



**Figura 67: Pantalla con las opciones de la app de gestión académica**

**Nota: Elaboración propia**

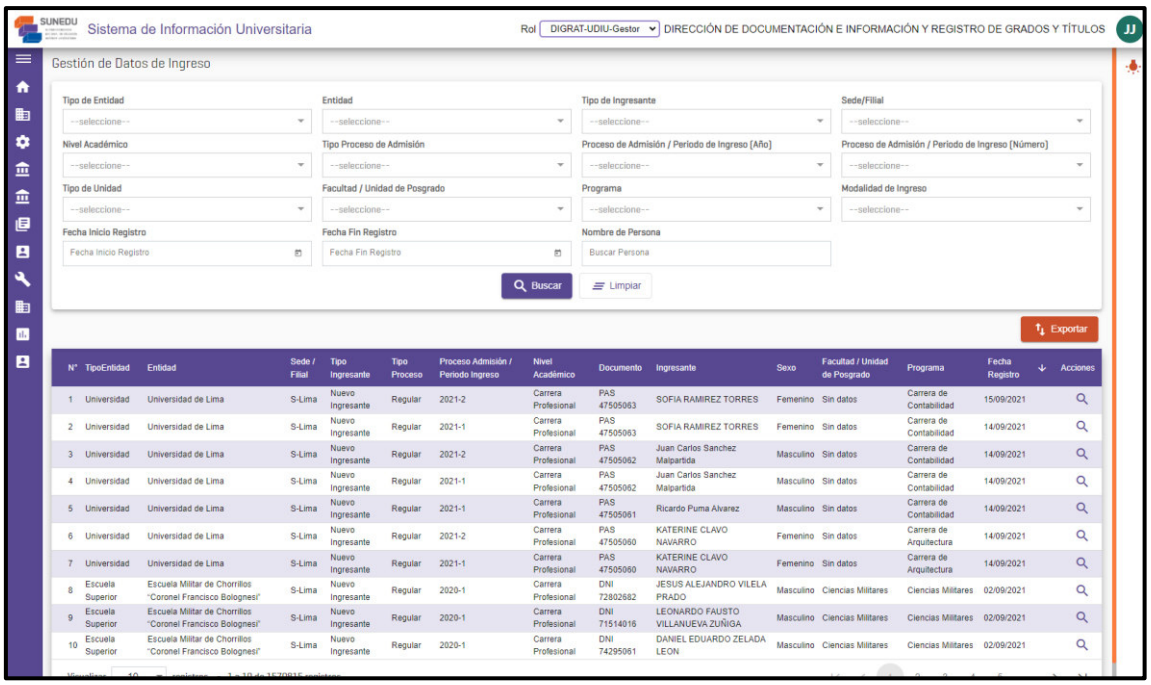



**Figura 68: Pantalla con la funcionalidad de Gestión de datos de ingresantes**

**Nota: Elaboración propia**

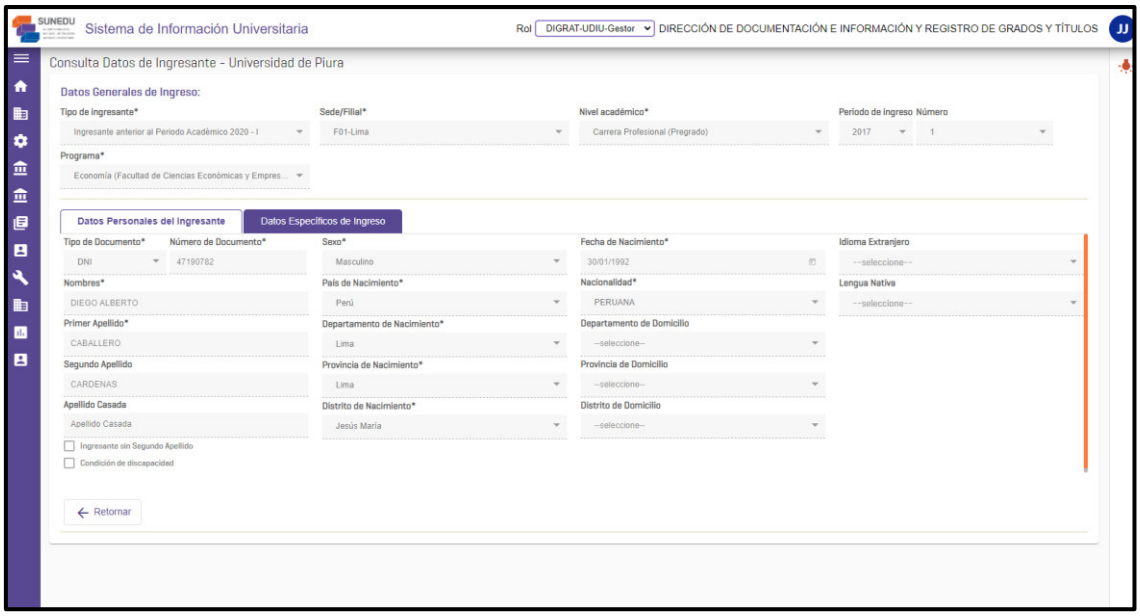



**Figura 69: Pantalla con la funcionalidad de Consulta de Datos de Ingresante**

**Nota: Elaboración propia**


**Figura 70: Pantalla con las opciones de la app de gestión curricular**

**Nota: Elaboración propia**

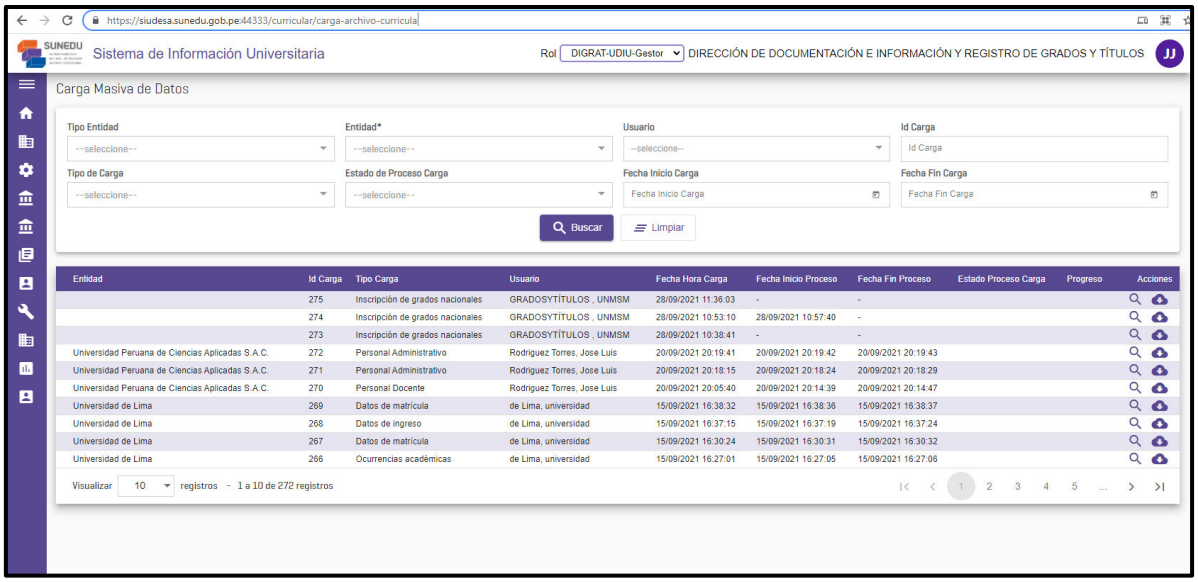



**Figura 71: Pantalla con la funcionalidad de Carga masiva de datos**

**Nota: Elaboración propia**

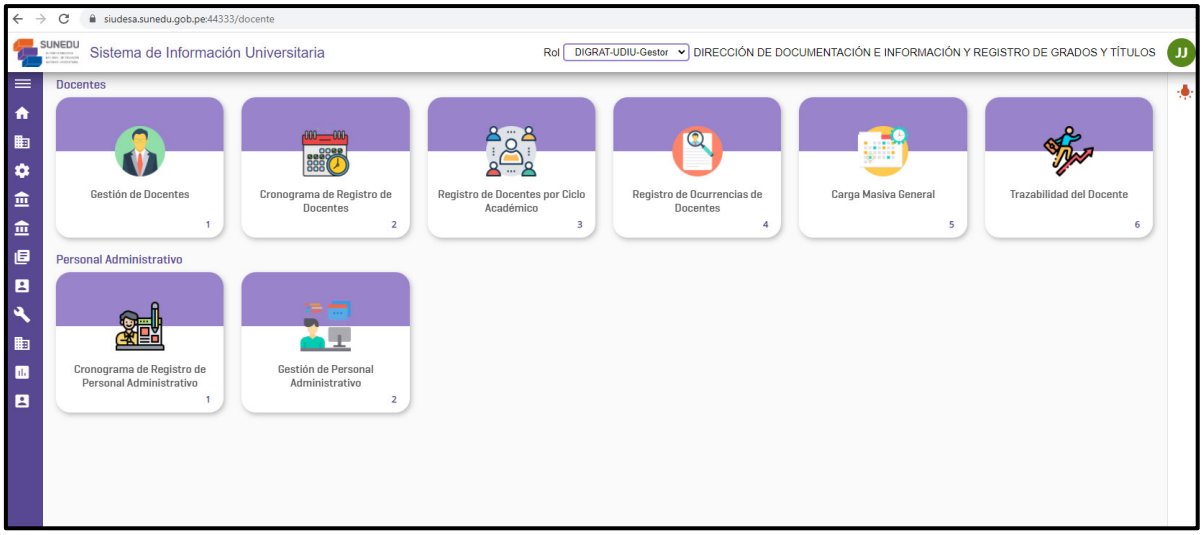



**Figura 72: Pantalla con las opciones de la app de docentes**

**Nota: Elaboración propia**

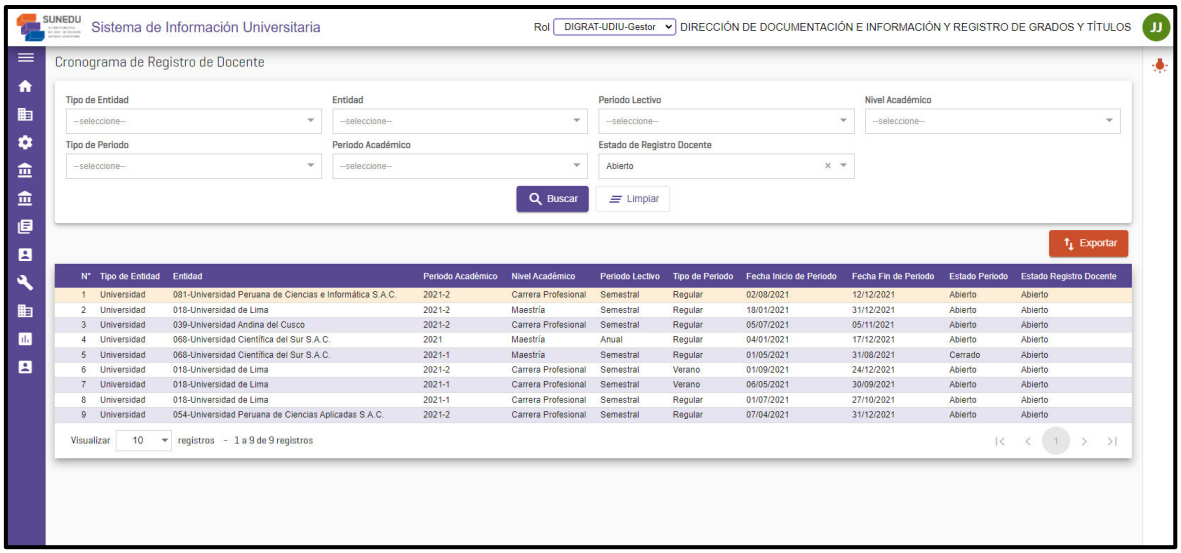



**Figura 73: Pantalla con la funcionalidad de gestión de registro de docentes**

**Nota: Elaboración propia**

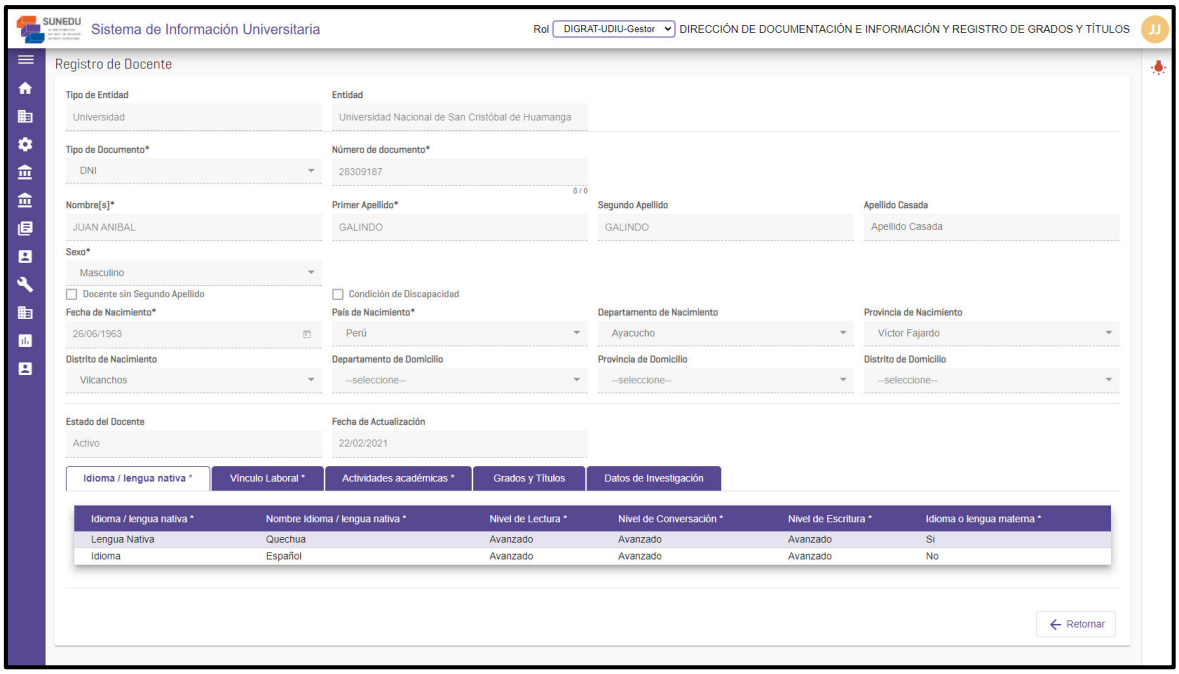



**Figura 74: Pantalla con la funcionalidad de registro de docente**

**Nota: Elaboración propia**

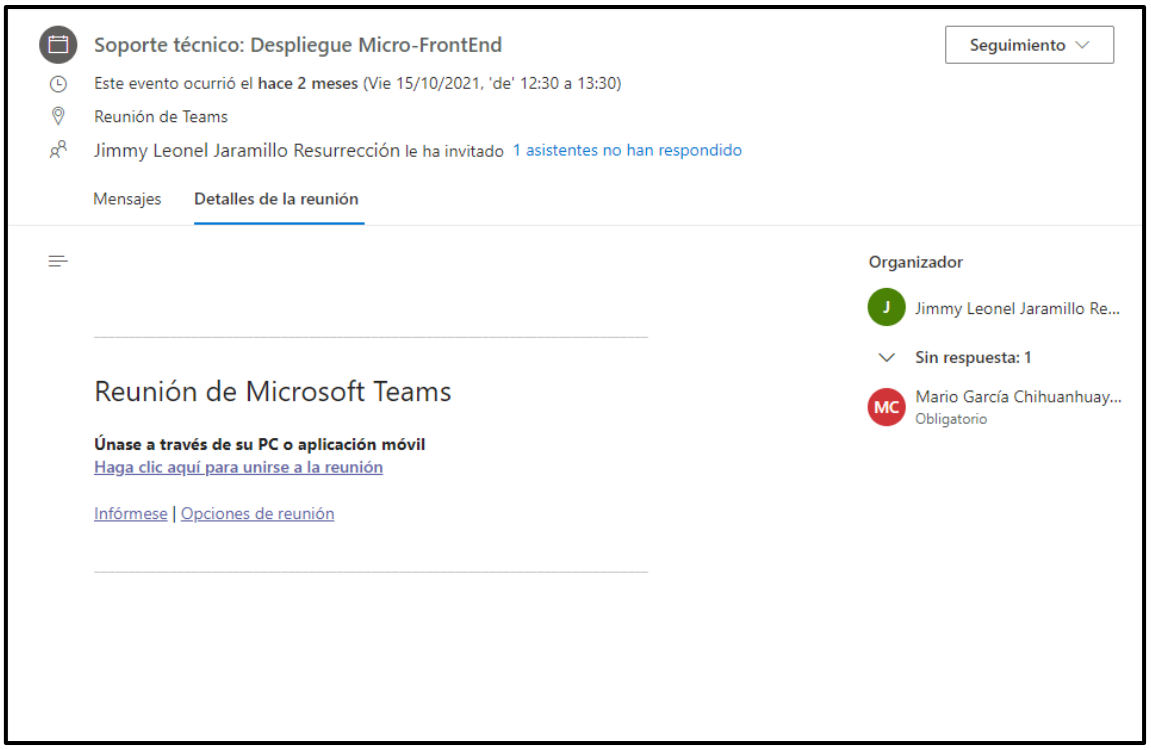



**Figura 75: Correo evidencia de reunión con el arquitecto para apoyo con el despliegue de la aplicación micro-frontend**

**Nota: Elaboración propia**

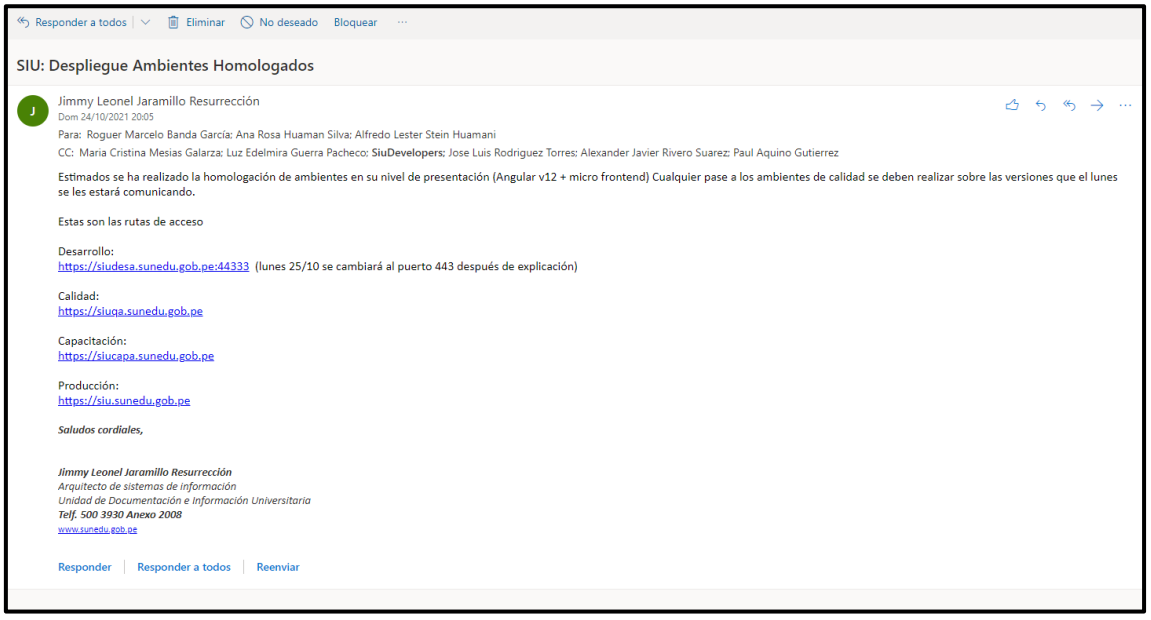



**Figura 76: Correo de evidencia del arquitecto del SIU sobre el despliegue de la aplicación micro-frontend en los diversos ambientes, incluido producción**

**Nota: Elaboración propia**

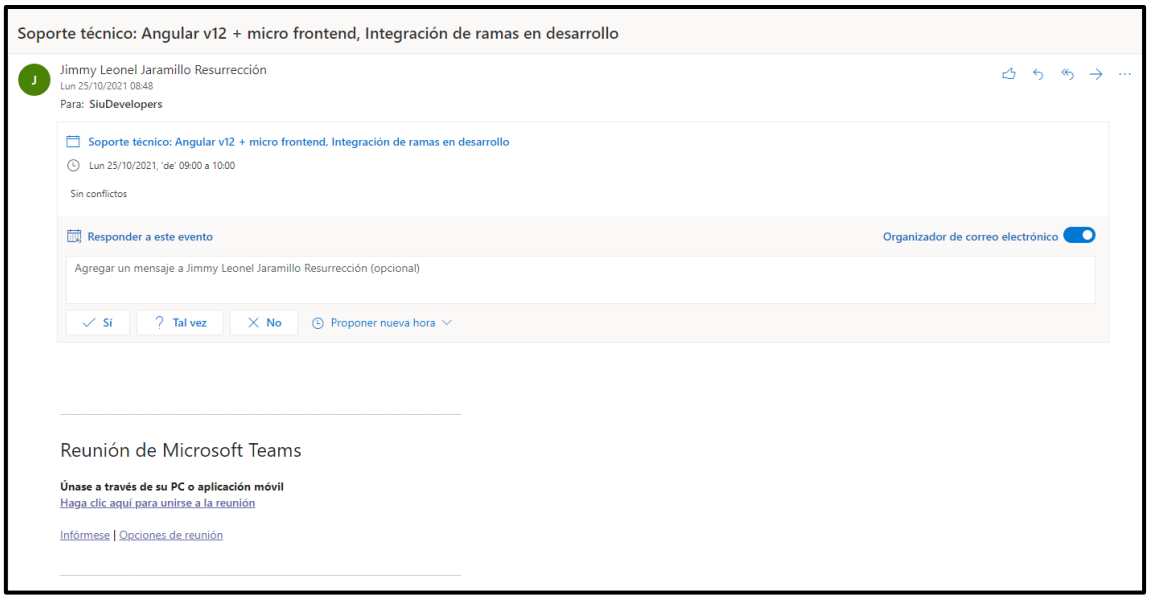



**Figura 77: Correo de evidencia del arquitecto del SIU para una reunión de capacitación al equipo del SIU sobre la nueva arquitectura micro-frontend**

**Nota: Elaboración propia**