



UNIVERSIDAD SIMÓN BOLÍVAR  
DECANATO DE ESTUDIOS PROFESIONALES  
COORDINACIÓN DE INGENIERÍA DE LA COMPUTACIÓN

**ASISTENTE PARA ENSAMBLAR APLICACIONES MÓVILES BASADO  
EN TECNOLOGÍA WEB**

Por:  
ALEJANDRO RENÉ GUILLÉN VERA

Realizado con la asesoría de:  
Tutor Académico: PROF. EDUMILIS MÉNDEZ  
Tutor Industrial: ING. ALEXANDER RAMÍREZ

**INFORME DE PASANTÍA LARGA**  
Presentado ante la Ilustre Universidad Simón Bolívar  
como requisito parcial para optar al título de  
Ingeniero en Computación

**Sartenejas, MAYO de 2016**



UNIVERSIDAD SIMÓN BOLÍVAR  
DECANATO DE ESTUDIOS PROFESIONALES  
COORDINACIÓN DE INGENIERÍA DE LA COMPUTACIÓN

**ASISTENTE PARA ENSAMBLAR APLICACIONES MÓVILES BASADO  
EN TECNOLOGÍA WEB**

Por:  
ALEJANDRO RENÉ GUILLÉN VERA

Realizado con la asesoría de:  
Tutor Académico: PROF. EDUMILIS MÉNDEZ  
Tutor Industrial: ING. ALEXANDER RAMÍREZ

**INFORME DE PASANTÍA LARGA**  
Presentado ante la Ilustre Universidad Simón Bolívar  
como requisito parcial para optar al título de  
Ingeniero en Computación

**Sartenejas, MAYO de 2016**

# **ASISTENTE PARA ENSAMBLAR APLICACIONES MÓVILES BASADO EN TECNOLOGÍA WEB**

Por:  
**ALEJANDRO RENE GUILLÉN VERA**

## **RESUMEN**

El presente informe describe las actividades realizadas durante el proyecto de pasantía larga, realizado en la empresa Synergy-GB, el cual consistió en diseñar y desarrollar un asistente que permita a la empresa ensamblar sus aplicaciones móviles, aprovechando los componentes reusables existentes del *framework* Kiraso, así como nuevos componentes que construya el desarrollador. Este asistente se desarrolló utilizando tecnología web de punta como Bootstrap y AngularJS y la integración con el *framework* Kiraso de Synergy-GB.

El asistente da al desarrollador dos vistas: una en la que es posible ensamblar gráficamente las aplicaciones con la parametrización respectiva de cada componente y otra en la que se puede editar el código fuente y se observa la vista previa de la aplicación en desarrollo. Por otro lado, también se muestra la actualización inmediata de la aplicación al realizar algún cambio en el código. Además, se pueden observar los mensajes de la consola que se producen durante los procesos de generación, construcción y ejecución de la aplicación.

Desde el punto de vista tecnológico se hizo uso de herramientas y tecnologías como JavaScript, TypeScript, AngularJS, NodeJS, Express, MongoDB, Socket.io, D3.js, Bower, NPM, Gulp, HTML5, Jade, Kiraso, Ionic y Yeoman.

# Índice General

Índice General	v
Índice de Tablas	ix
Índice de Figuras	x
Lista de Símbolos y Abreviaturas	xii
Introducción	1
1. Entorno Empresarial	4
2. Marco Teórico	7
2.1. <i>Framework</i> . . . . .	7
2.2. Metadatos . . . . .	8
2.3. Servicio web . . . . .	8
2.4. Transferencia de Estado Representacional . . . . .	8
2.5. <i>Single Page Application</i> . . . . .	9
2.6. <i>Websocket</i> . . . . .	9
2.7. Patrón de diseño . . . . .	9
2.8. Patrón MVC (Modelo-Vista-Controlador) . . . . .	9
2.9. Inyección de Dependencias . . . . .	10
2.10. Patrón Observador . . . . .	10
3. Marco Tecnológico	11
3.1. JavaScript . . . . .	11
3.2. TypeScript . . . . .	12
3.3. AngularJS . . . . .	12

3.4. NodeJS . . . . .	12
3.5. Express . . . . .	12
3.6. MongoDB . . . . .	13
3.7. Socket.io . . . . .	13
3.8. D3.js . . . . .	13
3.9. Bower . . . . .	14
3.10. NPM(Node package manager) . . . . .	14
3.11. Gulp . . . . .	14
3.12. Jade . . . . .	14
3.13. HTML . . . . .	14
3.14. SASS . . . . .	15
3.15. Kiraso . . . . .	15
3.16. <i>Ionic Framework</i> . . . . .	15
3.17. Yeoman . . . . .	15
3.18. JSON . . . . .	16
<b>4. Marco Metodológico</b>	<b>17</b>
4.1. Metodología de proyectos Synergy-GB . . . . .	17
4.1.1. Concepción . . . . .	17
4.1.2. Diseño . . . . .	18
4.1.3. Construcción . . . . .	18
4.1.4. Transición . . . . .	18
4.2. Metodología Interna . . . . .	19
4.3. Scrum . . . . .	19
4.3.1. El equipo Scrum (Scrum team) . . . . .	19
4.3.2. El Sprint . . . . .	20
4.3.3. Artefactos . . . . .	21
<b>5. Desarrollo del proyecto</b>	<b>22</b>
5.1. Concepción . . . . .	22
5.1.1. Requerimientos . . . . .	22
5.1.2. Riesgos . . . . .	24
5.1.3. Tecnologías y plataformas de desarrollo . . . . .	24
5.1.4. Planificación Inicial . . . . .	25

5.2. Diseño . . . . .	26
5.2.1. Arquitectura de Kiraso . . . . .	26
5.2.1.1. Generación de aplicaciones . . . . .	27
5.2.2. Definición de los componentes . . . . .	28
5.2.2.1. Tipos de componentes . . . . .	28
5.2.3. Diseño de la arquitectura . . . . .	29
5.2.3.1. Vista de Escenarios . . . . .	29
5.2.3.2. Vista Lógica . . . . .	30
5.2.3.3. Vista de Desarrollo . . . . .	30
5.2.3.4. Vista de Implantación . . . . .	31
5.2.3.5. Vista de Procesos . . . . .	31
5.2.3.6. Vista de Datos . . . . .	31
5.2.4. Planificación del desarrollo . . . . .	32
5.3. Construcción . . . . .	33
5.3.1. Iteración 1 . . . . .	33
5.3.2. Iteración 2 . . . . .	36
5.3.3. Iteración 3 . . . . .	38
5.3.4. Iteración 4 . . . . .	41
5.3.5. Descripción general de la aplicación . . . . .	42
5.4. Transición . . . . .	49
<b>6. Retos Enfrentados y Logros Adicionales</b>	<b>50</b>
6.1. Retos enfrentados . . . . .	50
6.2. Logros adicionales . . . . .	51
<b>7. Conclusiones y Recomendaciones</b>	<b>52</b>
<b>Bibliografía</b>	<b>54</b>
<b>Glosario</b>	<b>57</b>
<b>A. Lista de Riesgos</b>	<b>58</b>
<b>B. Casos de Uso</b>	<b>66</b>
<b>C. Documento de Arquitectura de Software</b>	<b>82</b>

D. Especificación de servicios del Asistente para ensamblar aplicaciones móviles	94
E. Matriz de pruebas	121
F. Manual Kiraso	138

# Índice de Tablas

5.1.	Resumen de requerimientos funcionales iniciales del Asistente para ensamblar aplicaciones móviles. . . . .	23
5.2.	Resumen de requerimientos funcionales adicionales del Asistente para ensamblar aplicaciones móviles. . . . .	23
5.3.	Resumen de requerimientos no funcionales del Asistente para ensamblar aplicaciones móviles. . . . .	24
5.4.	Resumen de los tipos de componentes. Elaborado por Synergy-GB. . . . .	28
5.5.	Esquema de proyectos. . . . .	32
5.6.	Esquema de usuarios. . . . .	32
5.7.	Iteraciones de desarrollo del Asistente para ensamblar aplicaciones móviles. .	33

# Índice de Figuras

1.1. Estructura Organizacional de Synergy-GB, C.A. . . . .	5
4.1. Etapas de la metodología de proyectos Synergy-GB. (Synergy-GB Área de Calidad) . . . . .	18
5.1. Stack de tecnologías utilizadas por Synergy-GB. Elaborado por Synergy-GB	26
5.2. Componentes necesarios para la generación de aplicaciones en Kiraso. Elaborado por Synergy-GB. . . . .	27
5.3. Diagrama de Casos de Uso del Asistente para ensamblar aplicaciones móviles.	29
5.4. Diagrama de Componentes de Asistente para ensamblar aplicaciones móviles.	30
5.5. Diagrama de Despliegue de Asistente para ensamblar aplicaciones móviles. .	31
5.6. Diseño vista gráfica. . . . .	34
5.7. Diseño vista previa. . . . .	34
5.8. Ejemplo de metadatos del componente lista de Synergy-GB. . . . .	39
5.9. Inicio de sesión del Asistente para ensamblar aplicaciones móviles. . . . .	43
5.10. Registro de usuarios del Asistente para ensamblar aplicaciones móviles. . . . .	43
5.11. Lista de aplicaciones de un usuario del Asistente para ensamblar aplicaciones móviles. . . . .	44
5.12. Formulario para crear nuevas aplicaciones del Asistente para ensamblar aplicaciones móviles. . . . .	44
5.13. Ejemplo de vista gráfica del Asistente para ensamblar aplicaciones móviles. Información del componente. . . . .	45
5.14. Ejemplo de vista gráfica del Asistente para ensamblar aplicaciones móviles. Información de los parámetros del componente. . . . .	45
5.15. Formulario para agregar un componente del Asistente para ensamblar aplicaciones móviles. . . . .	45

5.16. Ejemplo de vista gráfica del Asistente para ensamblar aplicaciones móviles. Aplicaciones pre-construidas. . . . .	46
5.17. Ejemplo de vista previa del Asistente para ensamblar aplicaciones móviles. .	46
5.18. Ejemplo de vista previa del Asistente para ensamblar aplicaciones móviles. Vista previa en dispositivo móvil. . . . .	47
5.19. Formulario para cargar recursos del Asistente para ensamblar aplicaciones móviles. . . . .	47
5.20. Descarga de archivo del Asistente para ensamblar aplicaciones móviles. . . .	48
5.21. Ejemplo de confirmaciones del Asistente para ensamblar aplicaciones móviles.	48

# **Lista de Símbolos y Abreviaturas**

**GB:** Global Business

**REST:** Representational State Transfer

**HTTP:** Hypertext Transfer Protocol

**MVC:** Modelo Vista Controlador (Model–View–Controller)

**DOM:** Document Object Model

**CSS:** Cascading Style Sheets

**D3:** Data-Driven Documents

**SVG:** Scalable Vector Graphics

**JSON:** JavaScript Object Notation

**SDK:** Software Development Kit

# Introducción

Synergy-GB es una empresa que desarrolla una gran variedad de productos y gran parte de su clientela está conformada por bancos, entre ellos Banco Mercantil, Banco Banesco, Banco Fondo Común (BFC), Banco Activo, entre otros. Además, también tienen clientes que provienen de otros sectores como el de telecomunicaciones.

## Planteamiento del problema

Actualmente, la demanda de soluciones a través del canal móvil aumenta día a día, y es cada vez más frecuente que una empresa considere ofrecer acceso a su oferta de servicios a través de las aplicaciones móviles. Hasta el momento, el desarrollo de estas aplicaciones demanda mucho tiempo debido a que se deben hacer desarrollos específicos por plataforma; sin embargo, ya es posible desarrollar dichas aplicaciones mediante tecnologías web sin sacrificar significativamente el desempeño.

Es por ello que Synergy-GB requiere de una herramienta que reduzca el esfuerzo necesario para cubrir dicha demanda mediante desarrollos híbridos usando tecnologías web, además de propiciar que el desarrollador trabaje de manera más estructurada y ordenada, de modo que sea fácil de mantener su producto. Por otro lado, al utilizar esta tecnología web, se busca reducir costos a la empresa, así como también reducir considerablemente los tiempos de desarrollo, manteniendo la calidad de los productos para continuar satisfaciendo las necesidades de sus clientes en el mercado nacional e internacional. Una de las principales razones por las que se busca este tipo de solución es debido a la alta rotación de los ingenieros o desarrolladores en las empresas venezolanas durante los últimos años.

El *framework* que utiliza Synergy-GB tenía el nombre de Megazord y fue el que se indicó en el Plan de Pasantías pero durante la ejecución del proyecto de pasantía la empresa tomó la decisión de cambiar su nombre a Kiraso por razones comerciales, es por ello que

cuando se haga referencia al *framework* se mencionará como Kiraso. Este *framework* facilita la generación de aplicaciones a partir de ciertos archivos de configuración, donde el usuario declara la forma y los componentes que van a formar parte de cada aplicación.

Desde el punto de vista arquitectónico, Kiraso se ubica en la capa superior dado que es una abstracción de las tecnologías comúnmente utilizadas por Synergy-GB para el desarrollo de aplicaciones híbridas multiplataforma.

## Solución propuesta

Se plantea el desarrollo de un asistente que permita:

- Ensamblar nuevas aplicaciones.
- Aprovechar los componentes existentes o nuevos que construya el desarrollador.
- Configurar y personalizar aplicaciones pre-construidas.

## Objetivos del proyecto

Todo lo anterior constituye el marco en el cual surge el presente proyecto de pasantía, el cual tiene por objetivo: “Desarrollar un asistente para ensamblar aplicaciones y personalizarlas basada en los componentes reusables y las herramientas provistas por el *framework* Kiraso de Synergy-GB”. En términos más concretos, los objetivos específicos del proyecto señalan lo siguiente:

- Levantar información sobre los requerimientos funcionales y de negocio del proceso de ensamblaje y generación de aplicaciones.
- Entender y documentar cómo funcionan las aplicaciones móviles definiendo un modelo general y en particular el *framework* Kiraso.
- Diseñar la arquitectura del asistente basándose en la arquitectura de Kiraso.
- Definir los metadatos que describen los componentes y aplicaciones que requiere el asistente.
- Definir la metáfora visual para configurar y personalizar aplicaciones pre-construidas y ensamblar nuevas aplicaciones que aprovechan los componentes existentes o nuevos que construya el desarrollador.
- Definir y desarrollar el asistente para ensamblar y generar aplicaciones.
- Desarrollar una arquitectura de micro servicios en la nube para almacenar los datos que consumen las aplicaciones.

- Desarrollar el asistente para cargar datos en la nube y sean servidos a través de la arquitectura de micro servicios en la nube.
- Definir el proceso de generación de aplicaciones en la nube.

## Alcance

Se definió como alcance del proyecto la entrega de un prototipo que cumpla con 100 % de funcionalidad. Los requerimientos no funcionales a cumplir son: interfaz sencilla, modularidad, que sea extensible y configurable. Su implantación no estuvo prevista como parte del proyecto de pasantía, así como los aspectos de seguridad inmersos cuentan como responsabilidad de la empresa. Los tipos de pruebas a aplicar por parte del pasante fueron funcionales a nivel de integración y de sistemas.

Este informe presenta los resultados del diseño e implementación del Asistente para ensamblar aplicaciones móviles basado en tecnología web. Se explicarán las diferentes fases del desarrollo del mismo y el proceso de transformación del concepto abstracto inicial en un prototipo funcional.

El informe está organizado de la siguiente manera: en el capítulo 1 se provee una visión general de Synergy-GB para familiarizar al lector con la misma. En el capítulo 2 se presentan algunos conceptos teóricos fundamentales. En el capítulo 3 se describen las herramientas tecnológicas utilizadas en el desarrollo del prototipo. En el capítulo 4 se describe la metodología de Synergy-GB utilizada en la pasantía. El capítulo 5 presenta el proceso completo de diseño y desarrollo del asistente. El capítulo 6 señala los retos enfrentados durante el desarrollo. Luego, se exponen las conclusiones y algunas recomendaciones derivadas del proceso de investigación y desarrollo, seguidas de las referencias bibliográficas y el glosario. Finalmente, en los Apéndices se presentan los artefactos que se realizaron a lo largo de este proyecto de pasantía.

# Capítulo 1

## Entorno Empresarial

En este capítulo se describe el ambiente organizacional en el que se desarrolló el Asistente para ensamblar aplicaciones móviles basado en tecnología web. Se presenta la empresa Synergy-GB, sus valores, objetivos y la estructura organizacional.

Synergy-GB es una empresa perteneciente al grupo Corporativo SYNERGY-GB Corporation, dedicada al desarrollo y comercialización de productos bajo Tecnologías de Información. Estudia las tendencias a nivel de aplicaciones corporativas actuales a las empresas, a fin de ofrecer soluciones en sus mercados que estén en línea con las prioridades gerenciales y de negocio del mundo actual.

La cartera de aplicaciones va desde Soluciones Integrales Sistémicas que resuelven una problemática compleja en la empresa, hasta Soluciones Puntuales Departamentales que resuelven problemas específicos en procesos de negocio donde se ha perdido el control gerencial [1].

La misión de la empresa, según revela su portal web es “crear, desarrollar y apoyar modelos de negocio que mejoren la competitividad y productividad de sus clientes” [1]. La visión de Synergy-GB es “Convertir a Synergy-GB en una organización de alcance hispanoamericano, que combine en forma creativa, ideas, talento, tecnología, visión empresarial, innovación y excelencia en el servicio” [1]. Los valores que definen a Synergy-GB como empresa son [1]:

- Compromiso con la Calidad.
- Compromiso con la Satisfacción al Cliente.
- Compromiso con la Generación de un Legado.
- Sentido de Propiedad.
- Sentido Emprendedor.
- Integridad y Honestidad.

- Orientación a Resultados.
- Compromiso con la Innovación y el Desarrollo de nuevas ideas.
- Proactividad.
- Trabajo en Equipo.
- Socialmente Responsables.
- Comercialmente Astutos.
- Diversión como elemento catalizador.

En la Figura 1.1 se presenta la estructura organizativa de la empresa [2]. El desarrollador ocupó el puesto de pasante en el área de aplicaciones.

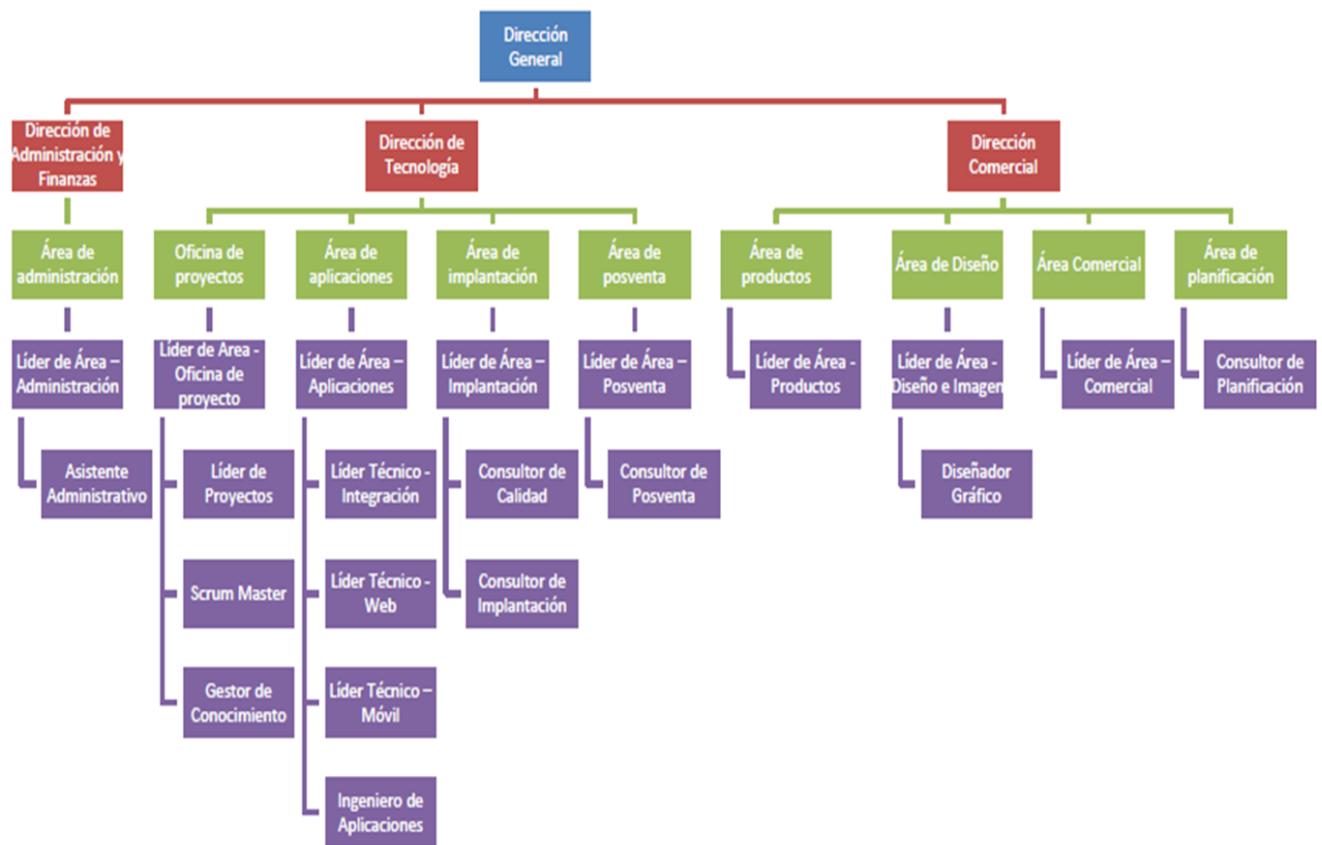


Figura 1.1: Estructura Organizacional de Synergy-GB, C.A.

Entre las áreas de la empresa, las siguientes son las más relevantes [3]:

- Oficina de proyectos:

Área dentro de la organización que se ocupa de centralizar y coordinar la dirección de proyectos.

- Diseño e Imagen:

Área encargada de realizar los diseños que proyecten la imagen Corporativa de la empresa y además de los diseños adecuados a la imagen de cada cliente de acuerdo a los objetivos de cada proyecto.

- Posventa:

Es el área que gestiona la relación con los clientes después que se implantan las soluciones, productos o servicios que ofrece la empresa, además se encarga de gestionar la atención de incidencias, errores, fallas, reclamos, mantenimiento, actualizaciones o requerimientos de nuestros clientes.

- Calidad e Implantación:

Área que se encarga de velar por la calidad de los servicios que ofrece la empresa de forma integral, desde los procesos, la metodología de trabajo y ejecución de proyectos hasta las encuestas de satisfacción después que se entrega una solución.

- Administración:

El área de Administración gerencia los procesos relacionados con la gestión de la información financiera y laboral de la empresa.

- Aplicaciones:

Esta área se encarga de gerenciar la fábrica de proyectos y productos en particular, la asignación del Talento a los diversos proyectos, productos o servicios de fábrica, implantación o posventa. El área desarrolla los Skills técnicos para el desarrollo de aplicaciones Móviles o Web así como la Integración de plataformas con foco continuo en la Innovación tecnológica y la entrega de soluciones de vanguardia.

- Unidad Comercial:

Esta unidad es la encargada de la gestión de ventas y relacionamiento con clientes y prospectos, coordinando las actividades que permitan llevar una oportunidad desde su detección (proactiva o reactivamente), hasta la presentación y negociación del cierre de la misma en el marco de una metodología que garantice una alta probabilidad de éxito.

- Unidad Productos:

Esta unidad es la máxima responsable de la gestión de productos de la organización. Su responsabilidad abarca desde la concepción de los productos hasta su maduración y consolidación. Gestiona los productos a lo largo de todo su ciclo de vida ayudando en cada momento a definir las estrategias comerciales y de mercadeo a seguir.

# Capítulo 2

## Marco Teórico

Para este desarrollo web se siguió el patrón de diseño observador, basado en eventos. Se utilizaron *frameworks* como AngulasJS, basado en patrones de diseño MVC y de inyección de dependencias, y Kiraso para generar las aplicaciones móviles utilizando un conjunto de metadatos. El desarrollo del Asistente se realizó de la forma más sencilla posible basándose en el concepto de *Single Page Application*. Además, se desarrolló un conjunto de servicios web REST para mantener la comunicación entre el *frontend* y el *backend*. Adicionalmente, se añadió la posibilidad de ver los subprocesos del servidor en la vista del cliente haciendo uso de *websockets*.

A continuación se describen los conceptos estudiados para el desarrollo de esta pasantía.

### 2.1. *Framework*

Un *framework* es el modelado de un dominio específico a través de un diseño abstracto. Específicamente en computación, se puede utilizar un *framework* (o varios) como base para el desarrollo de aplicaciones debido a que proporcionan a los desarrolladores una forma estructurada de trabajo. Esto trae como consecuencia que las aplicaciones o sistemas desarrollados utilizando un *framework* serán más fáciles de mantener si se sigue la arquitectura de *software* del mismo. [4].

## 2.2. Metadatos

Los metadatos son información estructurada que describen, explican, ubican o facilitan la extracción o el manejo de un recurso de información. Los metadatos son comúnmente llamados datos sobre datos o información sobre información [5].

## 2.3. Servicio web

Un servicio web es un conjunto de protocolos y estándares empleados para intercambiar información entre aplicaciones web. XML (Extensible Markup Language), SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language) y UDDI (Universal Description, Discovery, and Integration) son algunos de los estándares más comunes para lograr dicho intercambio. XML es utilizado para etiquetar los datos, SOAP para transferir la información, WSDL para describir los servicios disponibles y UDDI para listar qué servicios se encuentran disponibles para un momento dado [6].

También existen otros protocolos de servicios web como REST (*Representational State Transfer*), y distintos formatos de intercambio de datos y mensajes adicionales a XML tales como JSON (*JavaScript Object Notation*) [7].

## 2.4. Transferencia de Estado Representacional

La Transferencia de Estado Representacional (REST, por sus siglas en inglés), es un estilo arquitectónico para sistemas distribuidos. Los sistemas REST típicamente se basan en el protocolo HTTP para definir todas las operaciones que se pueden realizar. Estas operaciones aplican sobre recursos, los cuales son entidades semánticas que representan un objeto o un concepto bien definido dentro del sistema [7].

Entre los fundamentos de REST se encuentran [7]:

- Está basado en un protocolo cliente - servidor sin estado.
- Posee un conjunto de operaciones bien definidas que se aplican a todos los recursos de información.
- Ofrece una sintaxis uniforme para identificar los recursos.

## 2.5. *Single Page Application*

Una *Single-Page-Application* (o aplicación de una sola página), es una aplicación o sitio web que maneja una sola página web, con el objetivo de proveer una experiencia de uso más fluida al usuario. En una SPA se recupera todo el código necesario para cargar una sola página, o las fuentes necesarias se cargan dinámicamente y se agregan a la página conforme se necesiten [8].

## 2.6. *Websocket*

*Websocket* es un protocolo que habilita una comunicación bidireccional entre un cliente (comúnmente aplicaciones basadas en navegadores) y un servidor que no soporta múltiples conexiones HTTP. El protocolo consiste en la apertura con un protocolo de enlace, seguido del envío de mensajes, a nivel de la capa TCP [9].

## 2.7. Patrón de diseño

Un patrón de diseño describe la solución a un problema que ocurre una y otra vez en el ambiente de desarrollo para que de esta manera cada vez que aparezca el problema se utilice la misma solución [10].

## 2.8. Patrón MVC (Modelo-Vista-Controlador)

MVC es un patrón de diseño que busca estructurar aplicaciones interactivas de manera modular. Para ello se descompone la funcionalidad en tres componentes [11]:

- **Modelo:** Este componente encapsula las estructuras y funcionalidades de la aplicación. Esencialmente incluye el estado actual y las operaciones que lo pueden cambiar.
- **Vista:** Componente que a través de una interfaz gráfica de usuario presenta la información. Si el modelo es actualizado, la vista es notificada para solicitar la nueva información que se debe mostrar.
- **Controlador:** El propósito del controlador es desacoplar el modelo de la vista. Define la forma en la que la aplicación reaccionará a la interacción con el usuario.

## 2.9. Inyección de Dependencias

Es un patrón de diseño orientado a objetos que permite suministrar componentes (dependencias) a un objeto, en lugar de construirlos internamente. Cuenta con un ensamblador que se encarga de suministrar a tiempo de ejecución una implementación de la interfaz particular del componente [12]. Este patrón nos permite tener aplicaciones en las que esté completamente desacoplado el código fuente respecto a la implementación de algún componente externo que utilice.

## 2.10. Patrón Observador

Es un patrón de diseño orientado a objetos en el que se define una relación uno a muchos entre los objetos de un sistema, en el que cuando ocurre un cambio de estado en uno de ellos, todos sus dependientes son notificados y son actualizados inmediatamente [10]. Este patrón busca mantener comunicación entre los distintos componentes de un sistema los cuales sean completamente independientes entre sí.

# Capítulo 3

## Marco Tecnológico

En este capítulo se presentan las características principales de las herramientas tecnológicas seleccionadas para el desarrollo del proyecto de pasantía.

Para el desarrollo de este Asistente se utilizó el lenguaje de programación JavaScript y su notación de objetos (JSON). Se manejó el *stack* de MEAN.js, es decir, MongoDB para la base de datos, NodeJS y Express para exponer los servicios del *backend* y AngularJS para el *frontend*. Se utilizaron librerías como Socket.io para mantener la comunicación directa entre el cliente y el servidor, así como también D3.js para la construcción de grafos. El desarrollo se fundamentó en el *framework* Kiraso, que a su vez se basa en el *framework* Ionic. Kiraso utiliza TypeScript para la descripción de los archivos de metadatos. Al ser un proyecto de desarrollo web se usaron herramientas tales como Jade, HTML y SASS; además, se utilizaron herramientas de optimización como Gulp, Yeoman, Bower y NPM.

### 3.1. JavaScript

JavaScript es un lenguaje de programación interpretado orientado a objetos. Es usualmente conocido como lenguaje de *script* para páginas web, pero puede ser utilizado libremente en otros entornos. Es un lenguaje multiparadigma, dinámico y soporta estilos de programación imperativo, funcional y orientado a objetos [13].

## 3.2. TypeScript

TypeScript es un lenguaje desarrollado por Microsoft que agrega tipos, clases y módulos a JavaScript. Es un superconjunto tipado de JavaScript que permite escribir código de una forma más organizada. Es compilado a JavaScript legible y estandarizado [14].

## 3.3. AngularJS

Es un *framework* estructural para crear aplicaciones web dinámicas. Facilita el trabajo de los desarrolladores permitiendo la inyección de dependencias. Utiliza HTML extendido para proveer al desarrollador de nuevas funcionalidades. Funciona del lado del cliente únicamente, lo que da flexibilidad de poder usar cualquier tecnología del lado del servidor. Un uso adecuado del *framework* proporciona algunas ventajas como reusabilidad, modularidad, independencia del servidor y desacoplamiento entre los elementos del DOM y la lógica de la aplicación [15].

## 3.4. NodeJS

Node es una plataforma construida sobre V8 JavaScript Runtime, el cual es el motor de JavaScript de código abierto de Google (*V8 JavaScript Engine*, 2008), para construir aplicaciones de red escalables. Node utiliza un manejador de eventos y un modelo no bloqueante E/S que lo hace ligero y eficiente, ideal para aplicaciones de tiempo real que manejan gran cantidad de datos [16].

## 3.5. Express

Es un *framework* de desarrollo de aplicaciones web minimalista y flexible para NodeJS. Entre sus principales virtudes, podemos decir que es robusto, rápido, flexible y muy simple. Entre las características más importantes y potentes de Express tenemos los *middleware* [17]. Un *middleware* se puede definir como un *software* que asiste a una aplicación para comunicarse con otras aplicaciones, redes, *software* y/o sistemas operativos [18].

### 3.6. MongoDB

MongoDB es una base de datos, no relacional, ágil, basada en esquemas que permite cambiar rápidamente a medida que las aplicaciones evolucionan, además de poseer las características básicas de las bases de datos tradicionales [19].

Específicamente en este proyecto, se usó Mongoose.js. Este es un módulo de NodeJS que facilita el modelado de objetos en entornos asíncronos, dando conexión directa con MongoDB [20].

### 3.7. Socket.io

Socket.io es una herramienta que proporciona una comunicación bidireccional a tiempo real basada en eventos. Siempre buscando confiabilidad y velocidad, Socket.io funciona en cualquier navegador, dispositivo o plataforma. Comúnmente es usado para mantener estadísticas a tiempo real o mensajería instantánea [21].

Socket.io se basa sobre el protocolo de *Websockets*, el cual fue definido en el marco teórico del presente informe.

### 3.8. D3.js

D3 es una librería de JavaScript para manipular documentos basados en datos. D3 ayuda a darle vida a los datos usando HTML, SVG y CSS. El énfasis de D3 en los estándares web ofrece todas las capacidades de los navegadores modernos sin estar atado a *frameworks* propietarios, que combina poderosos componentes de visualización y enfoque basado en la manipulación de datos del DOM. D3.js permite enlazar datos arbitrarios al DOM, y luego aplicar transformaciones de los datos al documento. Con un mínimo de *overhead*, D3 es sumamente rápido, soporta gran cantidad de datos y comportamientos dinámicos para interacción y animación. El estilo funcional de D3 permite la reutilización de código a través de una diversa colección de componentes y *plugins* [22].

### 3.9. Bower

Bower es un manejador de versiones, que puede administrar componentes que contienen HTML, CSS, JavaScript, fuentes e incluso imágenes. Bower no concatena ni reduce código, solo instala las versiones adecuadas de los paquetes necesarios con sus dependencias [23].

### 3.10. NPM(Node package manager)

NPM por sus siglas *Node package manager*, es un manejador de paquetes o módulos que permite a los desarrolladores compartir soluciones y facilita la reutilización de código. Permite especificar versiones e instalar las dependencias de los paquetes necesarios para la ejecución de un proyecto [24].

### 3.11. Gulp

Gulp es un sistema de construcción que permite automatizar tareas que resulten tediosas o que consuman mucho tiempo a la hora de llevar el flujo de trabajo del desarrollo [25]. Algunas de estas tareas son la minificación de código JavaScript, recarga del navegador, compresión de imágenes, validación de sintaxis de código, entre otras.

### 3.12. Jade

Jade es un motor de plantillas de alto rendimiento, fuertemente influenciado por Haml (HTML Abstraction Markup Language), e implementado con Javascript para ser usado por NodeJS y/o navegadores [26].

Jade representa un lenguaje abreviado que facilita y agiliza escribir plantillas HTML.

### 3.13. HTML

El lenguaje de marcado de hipertexto, o HTML (*HyperText Markup Language*), es un lenguaje utilizado para crear y mostrar la representación visual de una página web. Este lenguaje solo sirve para determinar el contenido de una página web, no su funcionalidad. Para elaborar un documento web, HTML describe un conjunto de etiquetas y la estructura

que deben seguir [27]. Para crear sitios web dinámicos, HTML puede usarse muy fácilmente en conjunto con otras tecnologías tales como CSS y Javascript.

### **3.14. SASS**

SASS (*Syntactically Awesome StyleSheets*) es un preprocesador de CSS que ayuda a escribir código más complejo, de manera más natural para el desarrollador y facilita la mantenibilidad del mismo, además de proporcionar funcionalidades que CSS por si sólo no contempla [28].

### **3.15. Kiraso**

Kiraso es un *framework* desarrollado por Synergy-GB para crear aplicaciones híbridas usando componentes reusables. Este *framework* trabaja sobre otros elementos del *stack* como *Ionic Framework*, AngularJS y Cordova. Para lograr esto, el *framework* utiliza archivos de configuración que sirven para definir la estructura de una aplicación y los componentes que utilizará, con su respectiva configuración. También permite definir el enrutamiento y configuración global de la aplicación.

### **3.16. Ionic Framework**

Ionic es un *framework* orientado a construir aplicaciones móviles híbridas. Este tipo de aplicaciones tiene algunas ventajas sobre las aplicaciones nativas. Entre estas la disminución considerable de los tiempos de desarrollo, además de ser multiplataforma. Ionic también viene con un conjunto de elementos y diseños que hacen que la presentación de las aplicaciones sea lo más parecido posible a las nativas que se suele estar acostumbrado [29].

### **3.17. Yeoman**

Yeoman es un generador basado en herramientas y *frameworks* para ayudar a los desarrolladores a construir proyectos de forma rápida. Para mejorar la construcción de una aplicación, el flujo de trabajo de Yeoman comprende tres tipos de herramientas: la generación

de archivos necesarios con la herramienta yo, una herramienta de contrucción como Gulp y un manejador de paquetes como Bower o NPM [30].

### **3.18. JSON**

JSON (JavaScript Object Notation) es un formato ligero de intercambio de datos de fácil comprensión. Está basado como un subconjunto del lenguaje de programación Javascript. Este formato se construye sobre dos estructuras: en una colección de pares nombre/valor y en listas ordenadas de valores [31].

# Capítulo 4

## Marco Metodológico

En este capítulo se describe la metodología utilizada para el desarrollo del proyecto, la cual es una metodología híbrida debido a que se basa en OpenUp de cara al cliente pero utiliza Scrum en la fase de construcción. Además, se describe brevemente el marco de trabajo utilizado para la gestión del desarrollo del Asistente para ensamblar aplicaciones móviles basado en tecnología web.

### 4.1. Metodología de proyectos Synergy-GB

Describe las herramientas y artefactos que facilitan la comunicación y alineación tanto interna (entre los integrantes del equipo de trabajo) como externa (de Synergy-GB hacia el cliente) en cuanto al seguimiento y progreso de los proyectos [32].

En la Figura 4.1 se muestran las etapas en las cuales se divide la metodología de proyectos Synergy-GB [32].

#### 4.1.1. Concepción

En esta etapa se definen los documentos de requerimientos y el acuerdo con el cliente, acerca de los entregables que se deben realizar en el proceso de ejecución del proyecto. Los involucrados en esta etapa son el líder de proyecto (Synergy-GB) y el cliente. La persona responsable de cada una de las actividades es el líder de proyecto [32].

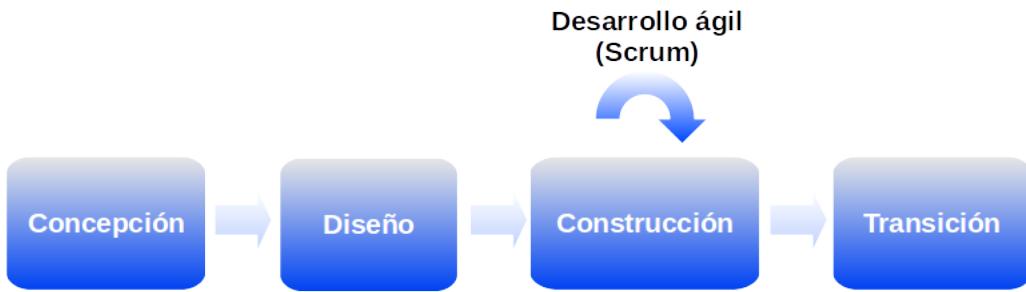


Figura 4.1: Etapas de la metodología de proyectos Synergy-GB. (Synergy-GB Área de Calidad).

#### 4.1.2. Diseño

En esta etapa se construyen los diseños de funcionalidad y servicios que se requieren para la etapa de construcción del producto. Las personas involucradas en esta etapa de la metodología son: líder de proyecto, arquitecto, diseñador, integración y calidad [32].

#### 4.1.3. Construcción

En esta etapa del proyecto, se realiza el desarrollo de la aplicación por parte del equipo del proyecto (Integración, Ux, Móvil, Web) encargado en el área correspondiente; en este caso, el encargado designado por la empresa es el pasante. Para su gestión, se utilizó el marco de trabajo Scrum. La fase se dividió en cuatro (4) iteraciones, de un mes de duración cada una. En el capítulo 5 se describirá con más detalle las actividades realizadas [32].

#### 4.1.4. Transición

La etapa de transición representa un punto de validación y cierre para el proyecto. Es a partir de este momento cuando se realizan todas las pruebas sobre los distintos componentes de la solución, determinando que su funcionamiento es el correcto y su presentación cumple con los diseños elaborados en tempranas etapas. Por lo tanto, se llevan a cabo las pruebas de estrés por parte del equipo de Integración y Ux, y luego las pruebas de los servicios e integrales por parte del equipo de calidad involucrado en el proyecto [32].

## 4.2. Metodología Interna

La metodología externa, entre Synergy-GB y el cliente sigue la metodología tal cual es descrita en OpenUp, sin embargo, la metodología utilizada internamente es un poco diferente. Se trabaja con las mismas fases descritas en OpenUp con la variante de que en la fase de la elaboración del producto, se utiliza la metodología de desarrollo ágil Scrum. La empresa Synergy-GB considera que esta es la forma más eficiente de trabajar con el equipo de desarrollo, algo que resulta transparente para el cliente.

## 4.3. Scrum

Es un marco de trabajo dentro del cual las personas pueden afrontar complejos problemas adaptativos, a la vez que entregan productos del máximo valor posible de forma productiva y creativa. No es una técnica de construcción de productos, sino simplemente un marco de trabajo flexible en donde se pueden emplear diversas técnicas o procesos [33].

Scrum se fundamenta en la teoría empírica de procesos, en donde se asegura que el conocimiento procede de la experiencia y de tomar decisiones basándose en lo que se conoce. Scrum emplea una aproximación iterativa e incremental para optimizar la predictibilidad y controlar el riesgo [33].

Al fundamentarse en la teoría empírica de procesos, Scrum debe soportarse sobre los siguientes tres pilares: transparencia, inspección y adaptación [33].

### 4.3.1. El equipo Scrum (Scrum team)

El Scrum team está conformado por el dueño del producto, el equipo de desarrollo y el Scrum master (jefe del Scrum o facilitador). A continuación, se especifican las responsabilidades de cada uno de los integrantes del equipo [33]:

- Dueño del producto (product owner):

Es el encargado de maximizar el valor del producto y del trabajo del equipo de desarrollo; a su vez, es la única persona que puede gestionar el *product backlog* (pila del producto). Es el responsable del producto desde un punto de vista comercial o de negocio.

- Equipo de desarrollo:

Es el equipo de trabajo que participa en un *sprint* dentro de la práctica Scrum. Realiza

las tareas de diseño y desarrollo de software. Desde un punto de vista más general, son los profesionales que realizan los incrementos de un producto, posiblemente funcional, al final de cada *sprint*. Sólo los integrantes del equipo de desarrollo están autorizados para realizar los incrementos del producto.

- Scrum master:

Tiene como responsabilidad asegurar que Scrum, como marco de trabajo, es entendido y llevado a cabo correctamente. Es un líder servil que se encuentra a la orden de todo el Scrum *team*. Por otra parte, el Scrum *master* es aquel puente entre agentes externos al Scrum *team* y este último.

#### 4.3.2. El Sprint

El núcleo de Scrum es el *sprint*, un período de tiempo de a lo sumo un mes durante el cual se crea un incremento de producto utilizable y potencialmente entregable. Cada nuevo *sprint* comienza inmediatamente después de la finalización del *sprint* anterior [33].

Para asegurar el correcto funcionamiento del *sprint*, deben seguirse las siguientes normas [33]:

- No se realizan cambios que afectarían al objetivo del *sprint*.
- La composición del Equipo de Desarrollo se mantiene constante.
- Los objetivos de calidad no disminuyen.
- El alcance puede ser clarificado y renegociado entre el Dueño del Producto y el Equipo de Desarrollo a medida que se va aprendiendo más.

Los *Sprints* contienen y consisten en la reunión de planificación del *sprint* (*Sprint Planning Meeting*), los Scrums diarios (*Daily Scrums*), el trabajo de desarrollo, la revisión del *sprint* (*Sprint review*), y la retrospectiva del *sprint* (*Sprint Retrospective*) [33]:

- Reunión de Planificación del *Sprint* (*Sprint Planning Meeting*):

El trabajo a realizar durante el *sprint* es planificado en la Reunión de Planificación de *sprint*. Este plan es creado mediante el trabajo colaborativo del Equipo Scrum al completo. La Reunión de Planificación de *sprint* consta de dos partes, cada una de ellas da respuesta a las siguientes preguntas, respectivamente: ¿Qué será entregado en el Incremento resultante del *sprint* que comienza? ¿Cómo se conseguirá hacer el trabajo necesario para conseguir el Incremento?

- Scrum Diarios (*Daily Scrums*):

Es una reunión de máximo 15 minutos, para que el Equipo de Desarrollo sincronice

sus actividades y cree un plan para las siguientes 24 horas. Esto se lleva a cabo inspeccionando el trabajo avanzado desde el último Scrum Diario y haciendo una predicción acerca del trabajo que podría ser completado antes del siguiente.

- Revisión del *sprint* (*Sprint Review*):

Al final del *sprint* se lleva a cabo una Revisión de *sprint*, para inspeccionar el Incremento y adaptar la Pila de Producto si fuese necesario. Durante la Revisión de *sprint*, el Equipo Scrum y los interesados colaboran acerca de lo que se ha hecho durante el *sprint*.

- Retrospectiva del *sprint* (*Sprint Retrospective*):

La Retrospectiva de *sprint* es una oportunidad para el Equipo Scrum de inspeccionarse a sí mismo, y crear un plan de mejoras que sean abordadas durante el siguiente *sprint*.

### 4.3.3. Artefactos

Los artefactos de Scrum representan trabajo o valor en diversas formas que son útiles para proporcionar transparencia y oportunidades para la inspección y adaptación. Los artefactos definidos por Scrum, están específicamente diseñados para maximizar la transparencia de la información clave, que es necesaria para asegurar que los Equipos Scrum tengan éxito al entregar un Incremento [33]:

- Pila de Producto (*Product Backlog*):

La Pila de Producto es una lista ordenada de todo lo que podría ser necesario en el producto, y es la única fuente de requerimientos para cualquier cambio a realizarse en el producto. El Dueño de Producto (*Product Owner*) es el responsable de la Pila de Producto, incluyendo su contenido, disponibilidad y ordenación.

- Pila de *Sprint* (*Sprint Backlog*):

La Pila de *Sprint* es el conjunto de elementos de la Pila de Producto seleccionados para el *sprint*, más un plan para entregar el incremento de producto y conseguir el objetivo del *sprint*.

Para los efectos de esta pasantía, el dueño del producto fue Alexander Ramírez. El *scrum master* fue Ana Dávila y el desarrollador fue el pasante designado por la empresa, supervisado por el tutor asignado Alexander Ramírez.

# **Capítulo 5**

## **Desarrollo del proyecto**

Este capítulo describe las actividades realizadas en la concepción, diseño y construcción del prototipo del Asistente para ensamblar aplicaciones móviles. Está dividido en secciones y a partir de cada una de ellas se representan las fases de la metodología adoptada descritas en el capítulo anterior.

### **5.1. Concepción**

El objetivo de esta fase consistió en recolectar la información necesaria para levantar los requerimientos del cliente para el proyecto a desarrollar. Por este motivo, se realizaron reuniones periódicas con el mismo, para determinar detalladamente dichos requerimientos y definir las características que el Asistente para ensamblar aplicaciones móviles debería tener. Esta fase tuvo una duración de 2 semanas, y se realizaron las siguientes actividades:

- Familiarización con la empresa y el entorno laboral.
- Investigación sobre las prácticas de la empresa.
- Levantamiento de requerimientos funcionales y no funcionales.
- Identificación y mitigación de riesgos que puedan afectar el desarrollo del Asistente.
- Estudio de las tecnologías a ser utilizadas para el desarrollo del proyecto.

#### **5.1.1. Requerimientos**

La lista inicial de requerimientos está basada en las solicitudes que el cliente ha realizado en las reuniones iniciales y en el planteamiento del proyecto propuesto por la empresa. Se

presenta en la Tabla 5.1 un resumen de los requerimientos funcionales iniciales del prototipo del Asistente para ensamblar aplicaciones móviles usando tecnología web.

Requerimiento	Descripción
<b>Construir aplicaciones nuevas</b>	Un usuario debe poder usar componentes existentes de la empresa o propios para poder construir una aplicación completamente nueva.
<b>Configurar y personalizar aplicaciones existentes</b>	Un usuario debe poder utilizar la estructura de una aplicación ya creada por él mismo o por la empresa y configurarla a la medida.
<b>Agregar nuevos componentes</b>	Un usuario debe poder utilizar componentes del <i>framework</i> Kiraso que haya desarrollado previamente.
<b>Generar archivos de configuración de Kiraso</b>	Un usuario debe poder generar los archivos de configuración necesarios para el <i>framework</i> Kiraso de modo que se pueda generar la aplicación con el <i>framework</i> .

Tabla 5.1: Resumen de requerimientos funcionales iniciales del Asistente para ensamblar aplicaciones móviles.

Es importante resaltar que durante el proceso de levantamiento de requerimientos, además de los requerimientos descritos anteriormente, se tomaron decisiones que serán manejadas como trabajo adicional y son representadas por los requerimientos funcionales adicionales de la aplicación que se presentan en la Tabla 5.2.

Requerimiento	Descripción
<b>Gestión de usuarios</b>	Un usuario debe poder iniciar sesión, cerrar sesión y crear una cuenta en la aplicación.
<b>Ensamblar aplicaciones gráficamente</b>	Un usuario debe poder crear una aplicación nueva de manera gráfica con la finalidad que sea fácil la vizualización de la estructura y jerarquía de la misma y de cada uno de sus componentes.
<b>Gestionar aplicaciones</b>	Un usuario debe poder crear nuevas aplicaciones, así como también, modificar sus aplicaciones existentes o eliminarlas.
<b>Gestionar estructura de archivos</b>	Un usuario debe tener la capacidad de ver en el Asistente la estructura de archivos de su aplicación en desarrollo, de igual manera, el usuario debe poder ver el contenido de los archivos dentro de esta estructura y modificarlos.
<b>Mostrar vista previa</b>	Un usuario debe poder ejecutar desde el Asistente su aplicación, haciendo uso del <i>framework</i> Kiraso y poder ver la vista previa de la misma desde el Asistente.
<b>Personalizar aplicaciones</b>	Un usuario debe poder darle estilo a sus aplicaciones usando los diseños que desea para que las aplicaciones generadas con el Asistente tengan presencia definitiva.

Tabla 5.2: Resumen de requerimientos funcionales adicionales del Asistente para ensamblar aplicaciones móviles.

Un resumen de los requerimientos no funcionales pueden verse reflejados en la Tabla 5.3.

Requerimiento	Descripción
<b>Mantenibilidad</b>	El Asistente debe ser desarrollado de forma modular y bien documentado, para facilitar y agilizar las labores de mantenimiento y/o actualización del <i>software</i> .
<b>Escalabilidad</b>	La aplicación debe manejar el crecimiento continuo de forma fluida, permitiendo agregar nuevos servicios o elementos fácilmente.
<b>Usabilidad</b>	La aplicación a desarrollar debe ser fácil de entender y usar, mostrando sus funcionalidades de forma intuitiva al usuario.

Tabla 5.3: Resumen de requerimientos no funcionales del Asistente para ensamblar aplicaciones móviles.

### 5.1.2. Riesgos

El propósito de la lista de riesgos es, en primer lugar, identificar aquellos elementos que pueden afectar negativamente la ejecución del proyecto. Además, permite medir el impacto de cada riesgo y planificar estrategias de mitigación y acciones de contingencia en caso de materializarse alguno de ellos.

Los principales riesgos detectados fueron, en su mayoría, desacuerdos que se originaban por problemas de comunicación entre el desarrollador y el cliente. Sin embargo, como en este caso el rol de cliente es asumido por la misma empresa Synergy-GB, las acciones tomadas para mitigar estos riesgos consisten en la realización de reuniones periódicas para mantener los objetivos y requerimientos claros y actualizados. Otro riesgo que influyó en el desarrollo de la aplicación fue asociado al cálculo erróneo del tiempo de desarrollo (para mayor detalle de los riesgos consultar el Apéndice A).

### 5.1.3. Tecnologías y plataformas de desarrollo

El proyecto debía ser desarrollado en ambientes que permitieran una buena flexibilidad y compatibilidad con las tecnologías más recientes para el desarrollo web. Esto implica el uso de herramientas básicas como HTML5 y CSS como estándares para el desarrollo de la interfaz, y el uso de un navegador compatible con los mismos. Por esta razón, el sistema fue desarrollado usando Google Chrome como navegador principal. Para la implementación del Asistente para ensamblar aplicaciones móviles se utilizaron las siguientes herramientas:

- Construcción del proyecto: Gulp.
- Manejo de dependencias: NPM y Bower.

- *Frontend:*
  - Lenguaje de programación: JavaScript
  - *Frameworks:* AngularJS, Bootstrap
  - Lenguaje de marcado: HTML
  - Motor de plantillas: Jade
  - Librerías: D3.js, FileSaver.js, angular-schema-form, angular-tree-control, ui-ace, lodash, socket.io
- *Backend:*
  - Lenguaje de programación: JavaScript
  - *Framework:* Express
  - Ambiente de ejecución: NodeJS
  - Manejador de base de datos NoSQL: MongoDB
  - Especificación de servicios: Swagger
  - Módulos de node: mongoose, lodash, fs, child\_process, mkdirp, nodemailer, express, multer, socket.io
- Generador de aplicaciones: Kiraso.

#### 5.1.4. Planificación Inicial

Esta actividad consistió en estimar una planificación para el desarrollo del proyecto: el orden en que serán implementadas las diferentes fases del proyecto y el tiempo que debía tomar cada una de ellas.

Se estableció un plan de cuatro iteraciones de 4 semanas cada una, aproximadamente. En la primera iteración se planificó hacer toda la interfaz gráfica del Asistente para ensamblar aplicaciones móviles. En la segunda iteración se decidió realizar toda la capa referente a los servicios REST que necesitaría consumir la aplicación o bien, el *backend*. En la tercera iteración se programó realizar la integración de los elementos de la interfaz gráfica con los servicios que debía consumir el Asistente para generar aplicaciones móviles. Finalmente, para la última iteración se planificó realizar pruebas de las aplicaciones generadas para ajustar detalles del Asistente para ensamblar aplicaciones móviles.

## 5.2. Diseño

En esta fase, inicialmente se estudió la arquitectura del *framework* Kiraso y todos sus componentes para posteriormente diseñar la arquitectura del Asistente para ensamblar aplicaciones móviles el cual se basa en la arquitectura de Kiraso y debe contemplar todas sus características. La arquitectura del Asistente ha sido diseñada junto con el dueño del producto. Esta fase tuvo una duración de dos semanas.

### 5.2.1. Arquitectura de Kiraso

Desde el punto de vista arquitectónico, Kiraso es la capa superior del *stack* de tecnologías utilizado. El *framework* procesa ciertos archivos de configuración con el objetivo de producir código propio de los otros *frameworks* ubicados en las capas inferiores del *stack*. Estos archivos de configuración son propiamente metadatos debido a que contienen información que es utilizada por el *framework* para producir nueva información.

En la Figura 5.1 se puede observar la ubicación de Kiraso en el *stack* de tecnologías utilizadas por Synergy-GB para el desarrollo de aplicaciones móviles híbridas.



Figura 5.1: Stack de tecnologías utilizadas por Synergy-GB. Elaborado por Synergy-GB

Los archivos de configuración permiten definir las siguientes características de una aplicación:

- Estructura de la aplicación y configuración de los componentes.
- Enrutamiento de la aplicación.
- Configuración general de la aplicación.
- Idiomas de la aplicación.

### 5.2.1.1. Generación de aplicaciones

A continuación se describirá el proceso de generación de aplicaciones de Kiraso. El *framework* a tiempo de elaboración (*Build Time*), apoyándose en los metadatos, genera cuatro archivos claves que son utilizados por la aplicación en tiempo de ejecución (*Run Time*):

- app.js: Este archivo contiene la configuración de la aplicación, específicamente los parámetros generales y el enrutamiento. La estructura de las aplicaciones construidas simulan la estructura de un grafo, donde cada nodo corresponde a un componente y los enlaces entre cada par de componentes corresponden al enrutamiento entre ellos.
- appScreens.js: En este archivo se encuentran todos los controladores de los componentes de la aplicación.
- custom.js: Este archivo contiene todos los *scripts* escritos específicamente para la aplicación. En este caso, el *framework* se encarga únicamente de crear el archivo, e incluir los *scripts* de forma que puedan ser ejecutados.
- i18n.js: En este archivo se encuentran los idiomas soportados por la aplicación con sus respectivas traducciones.

En la Figura 5.2 se muestran los distintos componentes que se requieren para generar una aplicación en Kiraso. El *framework* ya dispone de un manejador de eventos y un generador de aplicaciones. El desarrollador debe definir e incluir los recursos, metadatos, fuentes de datos y componentes que representen las distintas pantallas de la aplicación.



Figura 5.2: Componentes necesarios para la generación de aplicaciones en Kiraso. Elaborado por Synergy-GB.

### 5.2.2. Definición de los componentes

Los componentes básicamente son las distintas partes que conforman una aplicación. Algunos componentes se pueden visualizar explícitamente en la aplicación, ya que cuentan con un controlador y una interfaz. Hay otro tipo de componentes que no se pueden visualizar, estos tienen la función de definir el origen de los datos de otros componentes o de conectar un par de componentes. Las conexiones o transiciones entre los distintos componentes o estados, se logra a través del disparo de eventos. Un componente define el nombre de un evento cuya lógica es definida en los archivos de configuración (metadatos) de la aplicación, lo que permite desacoplar el enrutamiento de la misma.

Los componentes siguen el mismo patrón de diseño que AngularJS debido a que tienen la misma estructura. La vista se desarrolla en HTML5 y CSS3, mientras que la lógica en JavaScript.

Siguiendo el patrón de inyección de dependencias de AngularJS, los componentes hacen uso de ciertos servicios definidos en el *framework* Kiraso. Estas dependencias permiten la parametrización de dichos componentes para que puedan ser reutilizados en múltiples aplicaciones de distintos propósitos.

#### 5.2.2.1. Tipos de componentes

A continuación, se describirán los distintos componentes que se pueden desarrollar en el *framework* Kiraso.

Los componentes se diferencian según su funcionalidad. En la Tabla 5.4 se presenta un resumen de los diferentes tipos de componentes.

Tipo	Descripción
<b>Lógico / Interfaz de usuario (<i>Logic / UI</i>)</b>	Estos componentes proveen la lógica y la interfaz de usuario de alguna pantalla. Están compuestos por un controlador y una vista.
<b>Fuente de datos (<i>Data sources</i>)</b>	Tienen la función de definir el origen de los datos de los componentes. En general, los datos pueden provenir de algún servicio, de otro componente o pueden ser definidos directamente en los metadatos.
<b>Conectores de datos (<i>Data connectors</i>)</b>	Se encargan de transformar los datos entre componentes. En ciertos casos, el formato de la información suministrada por un componente difiere del formato de la información esperada por otro componente. Este tipo de componentes permiten modificar los datos para que exista concordancia entre los componentes lógicos.

Tabla 5.4: Resumen de los tipos de componentes. Elaborado por Synergy-GB.

### 5.2.3. Diseño de la arquitectura

A la hora de diseñar la arquitectura de esta aplicación, se tomó en consideración que entre sus características, la aplicación fuera escalable, mantenible y usable (puede detallarse en el Apéndice C).

#### 5.2.3.1. Vista de Escenarios

En la Figura 5.3 se presenta el Diagrama de Casos Uso del Asistente para ensamblar aplicaciones móviles, correspondiente a la Vista de Escenarios. En el Apéndice B se pueden observar en detalle los Casos de Uso de la aplicación.



Figura 5.3: Diagrama de Casos de Uso del Asistente para ensamblar aplicaciones móviles.

### 5.2.3.2. Vista Lógica

La Vista Lógica no fue desarrollada, dado que el desarrollo de la aplicación web sigue un estilo basado en eventos y no orientado a objetos.

### 5.2.3.3. Vista de Desarrollo

La vista de desarrollo de la arquitectura del Asistente para ensamblar aplicaciones móviles se ve reflejada en el Diagrama de Componentes que se muestra en la Figura 5.4

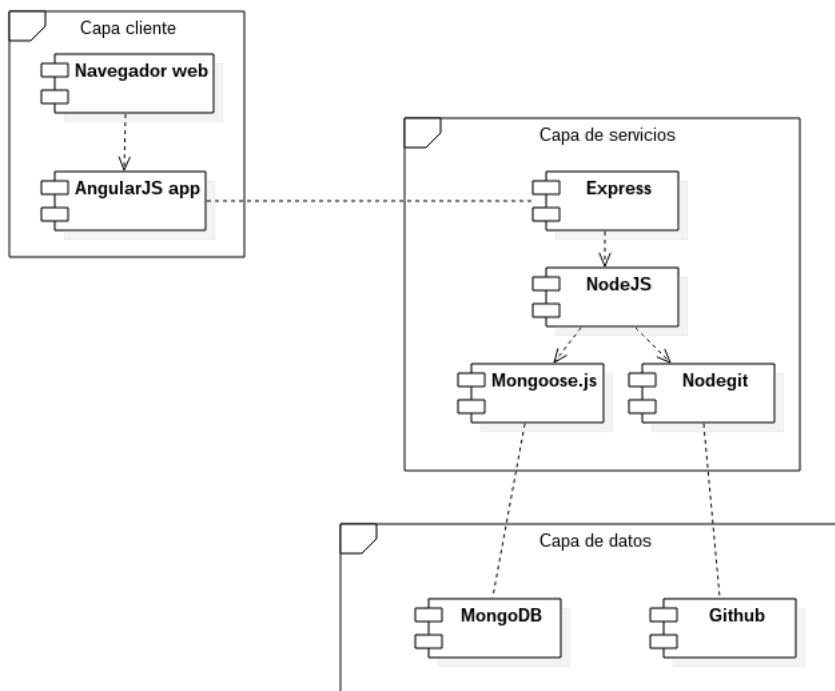


Figura 5.4: Diagrama de Componentes de Asistente para ensamblar aplicaciones móviles.

En este diagrama se puede observar el patrón de diseño MVC, el modelo está compuesto por los datos de la base de datos NoSQL MongoDB y datos provenientes de GitHub. El componente del servidor web representa al controlador y es el responsable a través de sus componentes internos de la interacción entre el modelo y la vista, la cual es representada a su vez por la aplicación web del Asistente para ensamblar aplicaciones móviles, encargada de presentar el modelo al usuario.

#### 5.2.3.4. Vista de Implementación

La vista de implantación de la arquitectura del Asistente para ensamblar aplicaciones se ve reflejada en el Diagrama de Despliegue que se muestra en la Figura 5.5

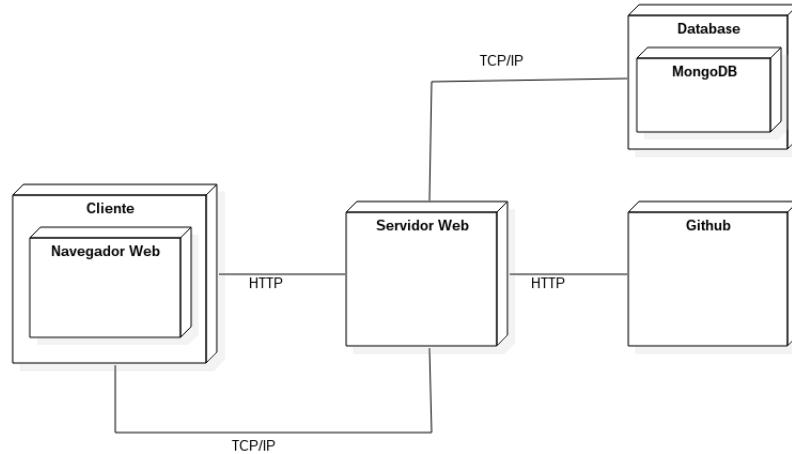


Figura 5.5: Diagrama de Despliegue de Asistente para ensamblar aplicaciones móviles.

#### 5.2.3.5. Vista de Procesos

La Vista de Procesos no fue desarrollada, dado que la aplicación no resuelve aspectos de concurrencia de usuarios y de datos.

#### 5.2.3.6. Vista de Datos

La aplicación se alimenta de los datos provistos por la base de datos NoSQL MongoDB, la cual se basa en esquemas clave/valor. Para este proyecto se definieron dos esquemas, uno que representa los usuarios y otro que representa los proyectos o aplicaciones móviles en desarrollo.

Para el esquema de los proyectos se tomó en cuenta en un principio únicamente dos atributos, el nombre de la aplicación, que ejerce como identificador del elemento y el grafo de la aplicación el cual contiene toda la información referente a la aplicación. Ambos elementos son del tipo *string*. El esquema se puede ver en la Tabla 5.5.

Por otro lado, el esquema de los usuarios posee 4 atributos. El nombre de usuario, el cual ejerce una función de identificador de los usuarios, su contraseña, la lista de los proyectos que

Clave	Tipo
<i>name</i>	<i>String</i>
<i>graph</i>	<i>String</i>

Tabla 5.5: Esquema de proyectos.

este tiene asociados y por último un atributo que indica si el usuario pertenece a la empresa o no. Este último, por los momentos, siempre tiene un valor por defecto que indica que los usuarios que se registran pertenecen a la empresa. Este atributo existe dada la idea de que la empresa en un futuro comercializará el *framework*, lo que llevaría a definir una lógica de negocio y determinar qué facilidades se le otorgarán a los futuros usuarios de esta aplicación que no pertenezcan a la empresa. El esquema se puede ver en la Tabla 5.6.

Clave	Tipo
<i>username</i>	<i>String</i>
<i>password</i>	<i>String</i>
<i>intern</i>	<i>Boolean</i>
<i>projects</i>	[ <i>String</i> ]

Tabla 5.6: Esquema de usuarios.

#### 5.2.4. Planificación del desarrollo

Para finalizar esta iteración se definió un plan inicial de desarrollo basado en los objetivos del proyecto, el plan de pasantía inicial y los requerimientos. Este plan consta de cuatro (4) iteraciones, de 4 semanas de duración cada una aproximadamente. En la Tabla 5.7 se describen los objetivos de cada iteración.

Iteración	Objetivos finales
Iteración 1	<ul style="list-style-type: none"> <li>■ Diseño detallado e implementación del <i>frontend</i> del Asistente para ensamblar aplicaciones móviles:           <ul style="list-style-type: none"> <li>• Vista de autenticación de usuarios.</li> <li>• Vista de ensamblaje de aplicaciones de manera gráfica.</li> <li>• Vista de previsualización de aplicaciones y edición de archivos.</li> </ul> </li> </ul>
Iteración 2	<ul style="list-style-type: none"> <li>■ Diseño detallado e implementación del <i>backend</i> del Asistente para ensamblar aplicaciones móviles:           <ul style="list-style-type: none"> <li>• Servicios de manejo de usuarios.</li> <li>• Servicios de manejo de sistemas de archivos.</li> <li>• Servicios de manejo de aplicaciones.</li> <li>• Servicios de manejo de componentes.</li> </ul> </li> </ul>
Iteración 3	<ul style="list-style-type: none"> <li>■ Integración de las funcionalidades implementadas en el <i>backend</i> con lo desarrollado en el <i>frontend</i>.</li> <li>■ Realizar ajustes necesarios.</li> </ul>
Iteración 4	<ul style="list-style-type: none"> <li>■ Ensamblar aplicaciones móviles utilizando el Asistente y realizar los ajustes necesarios.</li> <li>■ Finalización de los casos pendientes de las iteraciones anteriores.</li> </ul>

Tabla 5.7: Iteraciones de desarrollo del Asistente para ensamblar aplicaciones móviles.

## 5.3. Construcción

Esta fase se centró en el desarrollo de las funcionalidades y Casos de Uso previamente definidos, siguiendo la planificación especificada en la Tabla 5.7. El mismo se llevó a cabo de acuerdo a los estándares de Synergy-GB promoviendo la modularidad, reutilización y gerencia de cambio.

### 5.3.1. Iteración 1

La actividad inicial de esta iteración consistió en realizar los diseños del Asistente para ensamblar aplicaciones móviles basado en el *framework* Kiraso. Durante las reuniones que se realizaron en las primeras semanas de este proyecto se fue estructurando una idea de como iba a ser el Asistente. En las Figuras 5.6 y 5.7 se pueden ver algunos de los diseños iniciales.

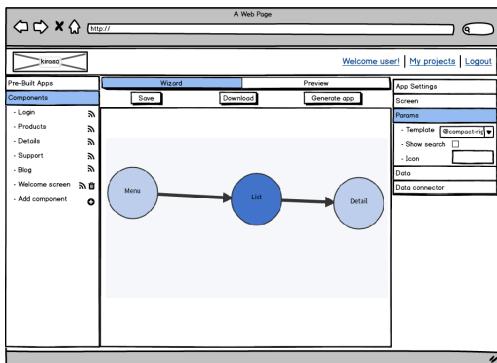


Figura 5.6: Diseño vista gráfica.

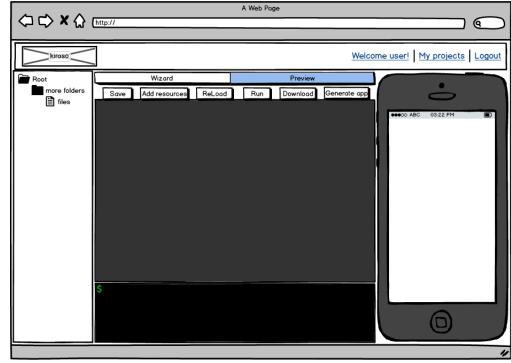


Figura 5.7: Diseño vista previa.

Dada la alta rotación de los desarrolladores en la empresa Synergy-GB, se decidió crear el *framework* Kiraso, el cual es una abstracción que facilita la creación de aplicaciones móviles, pero aún así, la empresa decidió que se necesitaba una herramienta para que la interacción con el *framework* fuese más fluida y ayudara a los desarrolladores a agilizar el proceso de creación de aplicaciones móviles. Es por ello que el diseño de este Asistente debía ser lo más simple posible dada la complejidad que ya agrega el hecho de tener que conocer las características de Kiraso. Para esto se decidió que la aplicación debería ser de la forma *Single Page Application*, de este modo, el desarrollador tendría acceso a todas las funcionalidades fácilmente sin tener que hacer transiciones.

Al desarrollar una aplicación móvil utilizando el *framework* sin el Asistente, el desarrollador debe definir un archivo de configuración en el cual se indica la estructura de la aplicación, definiendo un componente o pantalla “padre” y las pantallas que pertenecen a dicho componente, todos con sus respectivos parámetros de configuración. Al iniciar este proyecto, se quería que este proceso se pudiera visualizar de forma gráfica con la finalidad de facilitar su comprensión. Una vez que se estudió el funcionamiento de Kiraso y se entendió el proceso de creación de aplicaciones con el *framework*, se pudo observar que esta forma de estructurar una aplicación se podría visualizar fácilmente si el desarrollador construye un grafo dirigido. Los “padres” pueden representar los nodos raíz del grafo y los nodos siguientes representan las pantallas hijas. Por otro lado, los arcos simularían las rutas o transiciones entre pantallas.

Dadas las decisiones de tener una aplicación simple y que el proceso de ensamblaje de las aplicaciones móviles se realizara gráficamente, se tenía que dar a los desarrolladores la facilidad de escoger los elementos que formarían parte del grafo y a su vez la posibilidad de configurarlos. Para esto, se decidió tener un panel estático donde el usuario puede escoger los elementos que van a pertenecer a la aplicación y otro panel dinámico que cambiaría

dependiendo del componente o pantalla a configurar.

En las reuniones iniciales se llegó al consenso entre el pasante y el cliente que el alcance de este Asistente iría más allá de lo estipulado. Entre esas decisiones, estaba dar la opción al desarrollador de mostrar la vista previa de la aplicación pero además, se quería dar la posibilidad de ver y manipular directamente desde el Asistente los archivos de configuración de la aplicación. Si bien el Asistente facilitaría el trabajo, no se quería quitar libertad al desarrollador de aplicar sus conocimientos manipulando directamente la aplicación a desarrollar. Es por ello que manteniendo la simplicidad del Asistente, se adicionó una sección en la que se tiene un panel con la estructura de archivos de la aplicación móvil en desarrollo creada por el *framework* Kiraso, otro panel en el que se muestra la vista previa de la aplicación y una sección central en la que se tiene un editor de texto y una vista de los procesos que está llevando en ejecución el *framework* Kiraso.

Una vez que se terminaron los diseños y se llegó a un acuerdo, se procedió a empezar con el desarrollo de la interfaz del Asistente. Para ello, la empresa Synergy-GB dio al pasante una plantilla paga de un proyecto AngularJS, la cual proporcionó al pasante una estructura inicial que permitió agilizar el trabajo. Esta plantilla posee distintas características, como por ejemplo el uso de Jade el cual es un motor de plantillas que agiliza el proceso de desarrollo de las vistas usando HTML. Por otro lado esta plantilla utiliza Gulp para ejecutar algunas tareas de rutina y SASS como preprocesador de CSS. El uso de esta plantilla fue de gran ayuda, ya que permitió un desarrollo modular siguiendo la estructura del proyecto y se adecuó a las necesidades del desarrollo.

Para dar vida a la idea de ensamblar la aplicación de forma gráfica haciendo uso de un grafo dirigido, se utilizó un proyecto de código abierto para crear grafos de este estilo, el cual usa SVG y D3.js. Este proyecto fue ajustado a las necesidades del Asistente, como por ejemplo, cambiando la información necesaria de los nodos y arcos y modificando los procesos de creación del grafo y creación y eliminación de nodos. Además de esto, fue integrado completamente a la estructura de la aplicación, convirtiendo este proyecto en una directiva de AngularJS que se encarga del manejo del grafo.

En cuanto al panel de manejo dinámico de la configuración de los componentes, se utilizó una librería de AngularJS llamada angular-schema-form la cual permite fácilmente armar formularios dinámicos dado un esquema, las características del formulario y un modelo. La idea fue dar al Asistente la información necesaria para que cada componente tuviese su propio formulario dependiendo de su especificación. Además, para la construcción de los paneles de la vista gráfica se utilizaron elementos del módulo de Bootstrap para AngularJS.

Para la sección de la vista previa, se hizo uso del elemento de HTML Iframe el cual permite mostrar el contenido de una página web dentro de otra, de este modo se muestra en el Asistente la previsualización de la aplicación en desarrollo. Por otro lado, se utilizó Ace editor, el cual es un editor de texto web escrito en JavaScript y que posee un módulo para AngularJS llamado ui-ace. Este editor fue configurado para que tome en cuenta distintas sintaxis, las cuales aún son un poco limitadas, dependiendo de la extensión de los archivos a visualizar. Por último, para mostrar la estructura de archivos de la aplicación en desarrollo se utiliza un módulo de AngularJS llamado angular-tree-control el cual recibe el arreglo de elementos de la estructura de archivos y muestra los directorios y los respectivos archivos que contienen.

### 5.3.2. Iteración 2

En esta iteración se realizaron tareas relacionadas al diseño e implementación del *backend* del Asistente para ensamblar aplicaciones móviles. Los servicios a llevar a cabo fueron distribuidos de la siguiente manera:

- Servicios de manejo de usuarios.
- Servicios de manejo de sistemas de archivos.
- Servicios de manejo de aplicaciones.
- Servicios de manejo de componentes.

Para el desarrollo de los servicios que consumiría el Asistente, se utilizó NodeJS en conjunto con Express. Se hizo uso de la herramienta generadora de aplicaciones que provee Express llamada express-generator, la cual crea rápidamente el esqueleto de una aplicación. Este esqueleto garantiza modularidad en la implementación.

Para esta parte del desarrollo se tuvo que estudiar el funcionamiento de Express y en qué consistía su manejo de las rutas. Para el direccionamiento de las rutas se utilizó la clase express.Router que permite tratar las mismas de forma modular.

En esta fase se había estipulado el desarrollo de distintos tipos de servicios. Para cada uno de los tipos se estudió cuales serían las necesidades que tendrían que cubrir cada uno y se tuvo que realizar un estudio de las herramientas que proporcionaban las tecnologías seleccionadas para cumplir con dichas necesidades.

NodeJS nos proporciona por defecto un conjunto de librerías que podemos utilizar para cubrir distintos problemas. A lo largo de este desarrollo el módulo *File System* fue de los más utilizados, ya que proporciona funciones referentes a los archivos del sistema como por ejemplo, dar información de los archivos, manejo de permisología, lectura y escritura de

archivos, entre otras. De las funciones más utilizadas en este proyecto estuvo la lectura y escritura de archivos. Este módulo fue fundamental para los servicios referentes al manejo de sistemas de archivos.

Por otro lado, también se estudió la librería de NodeJS *Child Process*. Esta librería permite al desarrollador ejecutar distintos procesos. *Child Process* fue fundamental para estructurar los servicios que requerían la ejecución de los comandos del *framework* Kiraso.

A la hora de agregar los servicios referentes al manejo de usuarios y de la información de las aplicaciones en desarrollo, se decidió agregar MongoDB al proyecto. En reuniones que se llevaron a cabo con el Arquitecto de *Software* de la empresa, se hizo la recomendación al pasante de usar el módulo de NPM mongoose.js. Este módulo facilita las tareas de interacción con la base de datos y agiliza el trabajo del desarrollador. En esta fase se implementaron los esquemas para la base de datos definidos en los diseños.

En cuanto a los servicios referentes a los componentes, se necesitaba un módulo que permitiera la conexión de NodeJS con el sistema de control de versiones git, GitHub. Esto es debido a que la empresa trabaja con esta tecnología y se tomó la decisión de solo agregar componentes si su código fuente se encontraba en esta plataforma. Esta conexión es posible gracias al módulo de NPM llamado nodegit. Este módulo facilita la clonación de repositorios y el acceso a sus archivos.

Durante esta iteración se estudiaron principalmente las herramientas descritas anteriormente y se realizó una implementación de los servicios intentando que fueran lo más simple posible y pensando en los requerimientos del *frontend* del Asistente.

Los servicios desarrollados durante esta fase, contemplan el uso de métodos del protocolo HTTP. Básicamente se usaron los métodos GET para obtener información que requiera el Asistente y DELETE para eliminar algún componente o aplicación de la base de datos. La información necesaria que estos servicios requerían era enviada por parámetros en la solicitud. También se usaron los métodos PUT y POST para actualizar y agregar información respectivamente. Toda la información necesaria para estos métodos viene en el cuerpo del *request*.

Por otro lado, para controlar la estructura de archivos del lado del servidor se definieron ciertas reglas. En el directorio del servidor, se tendrá un directorio por usuario, que contendrá a su vez un directorio de la aplicación en desarrollo, un directorio temporal donde se guardan los archivos de configuración creados al cambiar a la vista de edición y vista previa, los cuales luego son trasladados al directorio original de la aplicación. Por último, un directorio *screens* donde se almacenan los componentes propios que haya agregado el usuario, donde

si luego son utilizados en una aplicación serán buscados en esta localización. Adicionalmente, al mismo nivel de los directorios de usuarios se define un directorio temporal que se utiliza a la hora de cargar los recursos al servidor para luego ser copiados a su respectiva ubicación en la aplicación. Si se quiere se puede ver la jerarquía de la siguiente manera:

```
/  
  /user  
    /app  
    /app_tmp  
    /screens  
  /tmp
```

En el directorio principal del servicio, a parte de los directorios de los usuarios habrá un directorio por componente de la empresa con su respectiva información de metadatos.

Swagger fue la herramienta utilizada para realizar la especificación de los servicios que fueron diseñados e implementados durante esta pasantía. Puede detallarse la especificación definitiva de los servicios implementados en el Apéndice D.

### 5.3.3. Iteración 3

Una vez iniciado el desarrollo del *frontend* y el *backend* del Asistente para ensamblar aplicaciones móviles se procedió a realizar la integración de ambas partes del proyecto para darle vida al Asistente.

El primer paso fue realizar la integración para el proceso de creación de aplicaciones. Para esto, se pasó por un proceso en el que se definieron ciertos datos necesarios de los que se alimentaría el Asistente. Entre estos tenemos unos archivos que hacen referencia al inventario de componentes que corresponden a la empresa y un archivo que hace referencia a los componentes que pertenecen al usuario que está desarrollando la aplicación. Esta información se define en un archivo JSON que contiene un arreglo de objetos con los siguientes atributos:

- Nombre del componente.
- Tipo de componente. Este atributo indica el componente que se está utilizando y es necesario, ya que es usado en la definición de los archivos de configuración de Kiraso.
- Dirección del repositorio del componente.

- Si es componente de la empresa, dirección en la que se encuentran los metadatos del archivo en el servidor.
- Atributo booleano que indica si el componente es propio de un usuario o no. Este atributo existe, ya que los componentes de los usuarios pueden ser eliminados.

Del mismo modo tenemos los archivos análogos que definen las aplicaciones preconstruidas que pueden ser utilizadas.

Después de haber definido estos archivos, se realizó la definición de los metadatos necesarios de los componentes para que los mismos puedan ser configurados a través del Asistente. En este caso se definen propiamente metadatos, ya que a partir de esta información, se genera el código necesario para mostrar los formularios dinámicos que proporciona angular-schema-form. Estos metadatos vienen de igual manera por un archivo JSON que define un conjunto de propiedades, entre las que se define un arreglo de parámetros que tienen los componentes, los cuales son utilizados para construir los formularios. Es importante mostrar que estos parámetros de los componentes deben seguir los tipos de datos que prevee angular-schema-form. En la Figura 5.8 se presenta un ejemplo de un archivo de metadatos de un componente.

```
{
  "name": "sgb-screen-list",
  "type": "screen",
  "params": {
    "param1": {
      "name": "showSearch",
      "type": "boolean",
      "description": "",
      "title": "Show search"
    },
    "param2": {
      "name": "templateType",
      "type": "string",
      "options": ["@compact-left", "@compact-right", "@large"],
      "description": "",
      "title": "Template type"
    },
    "param3": {
      "name": "showIcon",
      "type": "string",
      "description": "Use the name of a valid ionic icon",
      "title": "Icon"
    }
  }
}
```

Figura 5.8: Ejemplo de metadatos del componente lista de Synergy-GB.

Uno de los pasos más importantes en este proceso de ensamblar aplicaciones móviles utilizando el Asistente, consiste en la generación de los archivos de configuración que necesita el *framework* Kiraso para la construcción de la aplicación. Estos archivos son del tipo TypeScript y tienen toda la información referente a la estructura de la aplicación, así como la información necesaria para que los componentes de la aplicación funcionen correctamente.

Cada nodo o componente es un objeto de JavaScript que contiene toda la información referente a su configuración, la cual es almacenada cada vez que un usuario actualiza los datos

en el panel de formularios dinámicos. A su vez, cada vez que se selecciona un componente del grafo, se puede ver cual es la información que este tiene almacenado y decidir si actualizarla. Del mismo modo ocurre con los arcos, los cuales también almacenan la información referente a los eventos.

Cuando el usuario decide ir a la sección de vista previa de la aplicación en desarrollo, a partir de ese momento se generan los archivos de configuración correspondientes a Kiraso. En esa etapa del desarrollo se utilizó la herramienta lodash, que es una librería que permite la manipulación de objetos JavaScript. En primer lugar, se genera un archivo por componente el cual contiene toda la configuración respecto a tipos de componentes, fuente de datos, conectores de datos y parámetros de configuración del componente.

A continuación, el siguiente paso consistió en construir el archivo screens.ts, el cual es el que recibe Kiraso para construir la aplicación. Lo primero que se hizo fue realizar la importación de los archivos escritos anteriormente. En este archivo, la estructura de la aplicación viene dada por el nombre de cada pantalla, el cual debe ser precedido por el de su padre. De este modo, se tomó el grafo y se hizo una lista con los nodos raíz o padres, se construyó una matriz de adyacencias para poder establecer los nodos que son alcanzables desde cada raíz que se defina en la aplicación y de este modo poder construir correctamente el archivo de configuración. También se construyó en este proceso el archivo routes.ts donde se definen las rutas de la aplicación utilizando la información de los arcos del grafo.

Todas las solicitudes que se realizan al servidor trabajan de forma asíncrona. En este caso es importante que solo cuando se hayan resuelto todas las llamadas en las que se escriben estos archivos sea el momento de continuar con el flujo del programa, de otro modo cuando Kiraso vaya a construir la aplicación puede que haya información faltante. Para esto se ha utilizado un servicio que provee AngularJS llamado \$q el cual tiene un método que espera a todas las llamadas asíncronas y cuando terminen, continúa con la ejecución.

Es importante resaltar que durante esta iteración, hubo que modificar y expandir los servicios definidos en la iteración anterior. Una razón de esto pudo deberse a la falta de experiencia del pasante a la hora de diseñar los servicios y no tomar en cuenta la amplitud de situaciones que se debían abarcar.

Un caso importante a tomar en cuenta es que, al permitir que los desarrolladores de aplicaciones puedan modificar los archivos directamente en la sección de vista previa, había que mantener la información actualizada de los archivos de configuración con la información almacenada en los nodos que representaban al componente. Para ello, cuando se guardan los archivos de configuración se hizo un servicio solo para esta situación en la que se ejecuta un

método de Kiraso que compila la información del archivo en TypeScript y la lleva a un objeto JavaScript el cual es utilizado para actualizar la información del grafo.

Fue en esta iteración donde se introdujo al desarrollo la herramienta Socket.io. Esta herramienta permite la comunicación entre el Asistente y el servidor a través de *websockets*. Esta herramienta se utilizó en la sección de vista previa para que el desarrollador se mantenga informado de los procesos que está ejecutando el servidor, que a su vez, está ejecutando procesos del *framework* Kiraso.

Durante esta iteración también se agregaron las funcionalidades para descargar los archivos de la aplicación en desarrollo y la opción de cargar archivos para la personalización de las aplicaciones. En ambos casos, se tomó la decisión de trabajar con archivos comprimidos. En el caso de la carga de archivos, este comprimido puede ser del tipo zip o tar. Para poder ser recibido por el servicio se utilizó la herramienta multer, que es un módulo de NodeJS que permite la manipulación de archivos. Una vez recibido, el servicio implementado se encarga de descomprimir el archivo y agregar la carpeta que contiene los recursos al directorio *theme* de la estructura de archivos de Kiraso donde pueden ser utilizados para dar estilos a las aplicaciones.

### 5.3.4. Iteración 4

En esta iteración se procedió a construir aplicaciones utilizando el Asistente. A medida que se iban construyendo las aplicaciones se fueron arreglando detalles de la implementación.

Para ensamblar y construir aplicaciones móviles utilizando este Asistente, se utilizó el generador de Kiraso y el *framework* en sí. A la hora de ejecutar el generador de Kiraso, el usuario debe colocar el nombre de la aplicación por terminal cuando sea solicitado, pero esta manera de trabajar no era la más adecuada. Es por esto que, el pasante, agregó una nueva opción al generador de Kiraso, basado en Yeoman, la cual permite que el usuario pase el nombre de la aplicación como argumento, al ejecutar el comando de generación de Kiraso. De esta manera, el Asistente usa el nombre con el que ha sido creada la aplicación para la generación del proyecto. Adicionalmente, cuando se ejecuta por línea de comando la aplicación en desarrollo, el *framework* despliega inmediatamente una ventana en el navegador con la vista previa de la aplicación. Como en este caso se quiere llevar de manera controlada la vista previa de la aplicación en el Asistente y además no se puede permitir que en el servidor se ejecuten estos procesos de manera innecesaria, el pasante adicionó una opción al SDK del *framework* en la que, cuando se ejecuta el comando y recibe un *flag* específico,

no se despliega la vista previa de la aplicación en desarrollo en el navegador web, dejando la posibilidad al usuario de decidir en que momento visualizar la vista previa a través del Asistente.

Un trabajo importante durante esta iteración fue culminar algunos aspectos de la implementación que no habían sido cubiertos en las iteraciones anteriores. En primer lugar, se agregaron validaciones y confirmaciones a la aplicación. Es importante tomar en cuenta el error humano, por tanto cada vez que el usuario cambia de vista o desea salir del Asistente, se confirme que así lo desea. Por ejemplo, en la vista del ensamblaje gráfico de la aplicación, el usuario debe asegurarse de haber guardado el grafo actual, debido a que a partir de estos datos se procede a generarse los archivos de configuración, además es necesario almacenar esta información en la base de datos para poder recargar la aplicación en otra ocasión. Por otro lado, cuando el desarrollador se encuentra en la sección de vista previa y confirma que desea salir o desea ir a la vista gráfica, se elimina el *socket* que estaba abierto mandando información entre el *backend* y el *frontend* y se terminan los subprocesos en ejecución referentes a Kiraso.

Adicionalmente, con algunos consejos del equipo de diseño de la empresa, se mejoraron algunos detalles de la interfaz gráfica de la aplicación de modo que fuese más llamativa a la hora de interactuar con ella.

A lo largo de este desarrollo se definieron casos de prueba para comprobar la funcionalidad de la aplicación, la cuál cumple con un 100 % de funcionalidad. Estos se pueden ver en detalle en el Apéndice E.

Al final de esta iteración se realizó un manual sencillo que hace referencia a la instalación del ambiente del Asistente. En este también se expone cómo generar los archivos de metadatos y se presentan los metadatos de los componentes ya existentes de la empresa los cuales fueron definidos por el pasante. Este manual se encuentra en la plataforma que tiene Synergy-GB para documentar sus productos y se puede ver en detalle en el Apéndice F.

### **5.3.5. Descripción general de la aplicación**

Al finalizar la fase de construcción, se cuenta con un prototipo funcional del Asistente para ensamblar aplicaciones móviles validado por el cliente. El resultado de este desarrollo permite: iniciar y cerrar sesión, construir aplicaciones nuevas, usar aplicaciones preconstruidas, agregar componentes nuevos, ver vista previa de la aplicación, mostrar y editar archivos de la aplicación, personalizar aplicaciones y descargar archivos de la aplicación.

En la Figura 5.9 se muestra la pantalla de la página de inicio de sesión del Asistente.

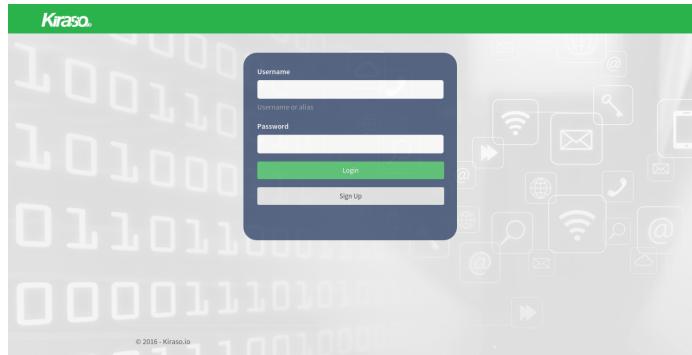


Figura 5.9: Inicio de sesión del Asistente para ensamblar aplicaciones móviles.

En la Figura 5.10 se muestra la pantalla de la página para registrar usuarios del Asistente.



Figura 5.10: Registro de usuarios del Asistente para ensamblar aplicaciones móviles.

Una vez iniciada la sesión del usuario se visualiza la página donde se listan los proyectos del mismo, además provee las funcionalidades para eliminar una aplicación, así como la posibilidad de crear una nueva. En la Figura 5.11 se muestra la captura de pantalla de esta página.

Se puede apreciar en la Figura 5.12 el formulario que se le presenta a un usuario para crear una nueva aplicación.

Una vez que el usuario selecciona o crea un proyecto es re-direccionado a la vista del Asistente donde puede estructurar gráficamente su aplicación. En esta vista puede seleccionar los elementos en el panel izquierdo que representan los componentes de Kiraso. Estos componentes al ser seleccionados se transforman en nodos del grafo. En la Figura 5.13 se muestra un ejemplo de esta vista.



Figura 5.11: Lista de aplicaciones de un usuario del Asistente para ensamblar aplicaciones móviles.

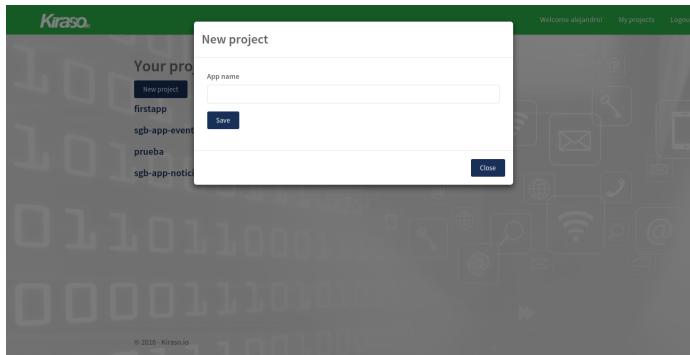


Figura 5.12: Formulario para crear nuevas aplicaciones del Asistente para ensamblar aplicaciones móviles.

Adicionalmente en la misma Figura 5.13 podemos observar otros aspectos, por ejemplo en el panel de componentes vemos que cada elemento tiene una opción que re-direcciona a la documentación de cada componente en GitHub. Por otro lado, los componentes que pertenecen al usuario en sesión tienen también la opción de eliminar sus componentes agregados.

Así mismo, podemos observar como el panel derecho muestra información referente al nodo que este siendo seleccionado en el grafo. En la Figura 5.13 se observa la selección del nodo header y se muestra en el panel derecho su información referente a los datos de la pantalla. En la Figura 5.14 se muestra seleccionado el nodo bonos-list y se muestra su información referente a los parámetros requeridos por la pantalla.

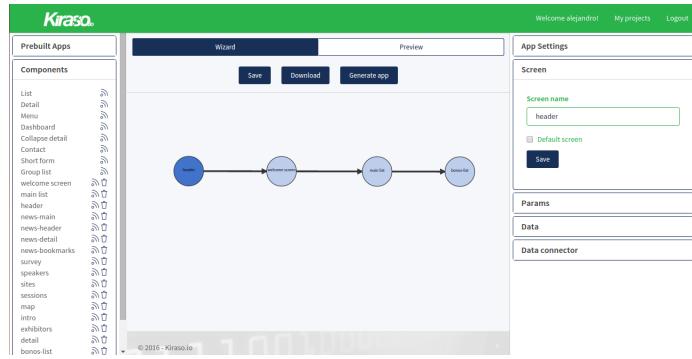


Figura 5.13: Ejemplo de vista gráfica del Asistente para ensamblar aplicaciones móviles. Información del componente.

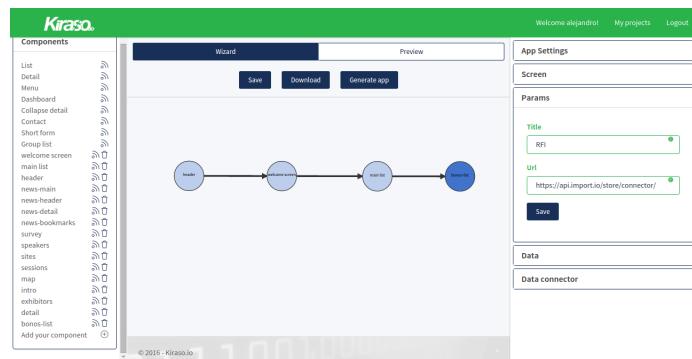


Figura 5.14: Ejemplo de vista gráfica del Asistente para ensamblar aplicaciones móviles. Información de los parámetros del componente.

En la Figura 5.15 se observa el formulario que se le presenta al usuario a la hora de dar la opción de agregar un componente a su lista.

This screenshot shows a modal dialog box titled 'Add your component'. It contains fields for 'Component name' (with 'RFI' entered), 'Component type' (with 'Ex: @rgb-screen-calendar (must have @)'), 'Github repo url' (with 'https://github.com/...'), and a 'Save' button. To the right of the dialog, the 'App Settings' panel is visible, showing the 'Screen' section with 'Screen name' set to 'header' and a checkbox for 'Default screen'. A 'Save' button is present. Below this are sections for 'Params', 'Data', and 'Data connector'.

Figura 5.15: Formulario para agregar un componente del Asistente para ensamblar aplicaciones móviles.

El Asistente también permite reusar la estructura de las aplicaciones que ya hayan sido construidas previamente. En la Figura 5.16 vemos en el panel izquierdo la sección de aplicaciones pre-construidas del usuario.

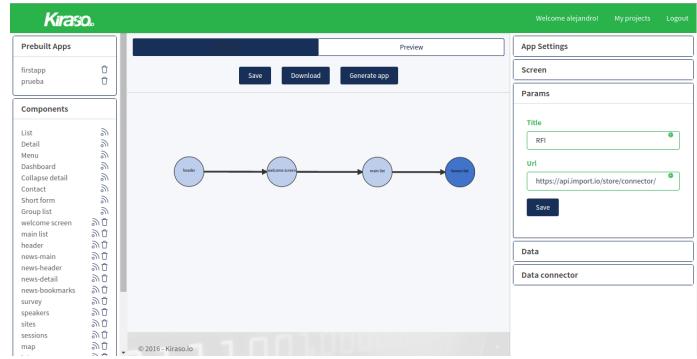


Figura 5.16: Ejemplo de vista gráfica del Asistente para ensamblar aplicaciones móviles. Aplicaciones pre-construidas.

Una vez que el usuario termina de configurar su aplicación gráficamente, puede proceder a la sección de vista previa del Asistente, la cual se muestra en la Figura 5.17.

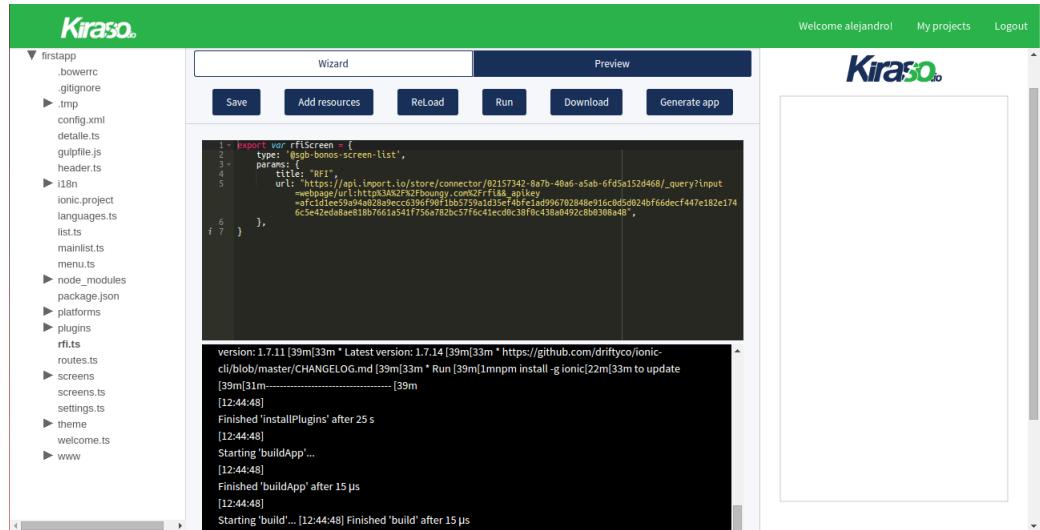


Figura 5.17: Ejemplo de vista previa del Asistente para ensamblar aplicaciones móviles.

En la misma Figura 5.17 podemos observar a la izquierda la estructura de archivos del proyecto Kiraso en desarrollo. En la sección central apreciamos el editor de texto web y el ejemplo de uno de los archivos seleccionados, así como los mensajes de todos los procesos que el servidor ejecuta. En la Figura 5.18 se muestra la vista previa de como se vería en un dispositivo móvil la aplicación en ejecución.

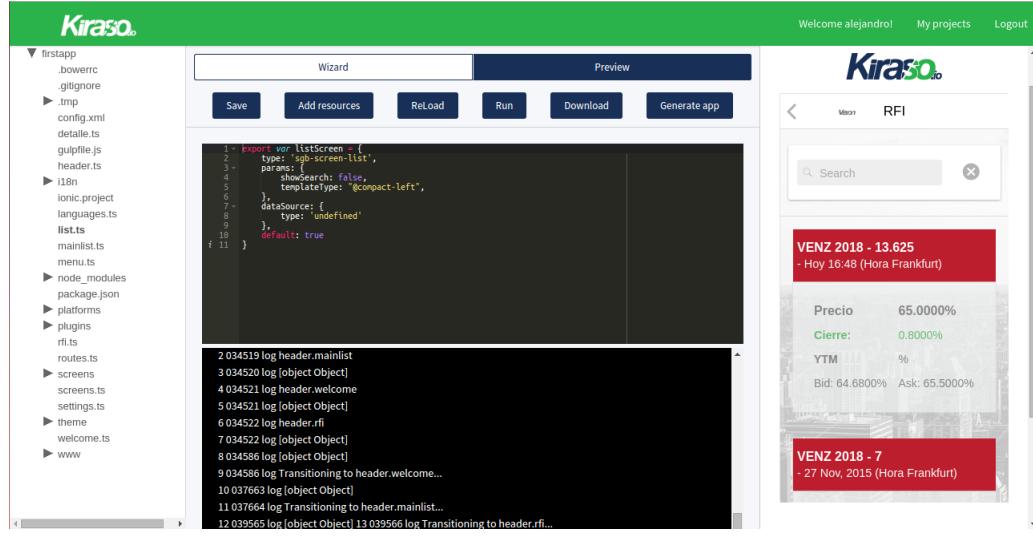


Figura 5.18: Ejemplo de vista previa del Asistente para ensamblar aplicaciones móviles. Vista previa en dispositivo móvil.

Cuando el usuario selecciona la opción de agregar recursos, se puede observar el formulario que se le muestra para cargar los mismos en la Figura 5.19.

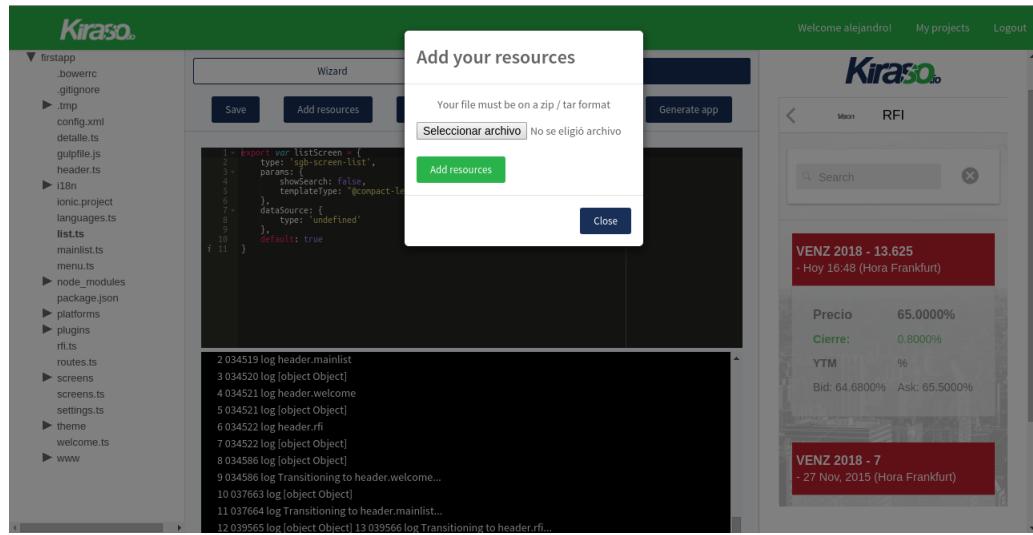


Figura 5.19: Formulario para cargar recursos del Asistente para ensamblar aplicaciones móviles.

Por otro lado, el usuario tiene la opción de descargar el proyecto en desarrollo en todo momento, para esto una vez que el usuario selecciona la opción de descarga se abre la ventana de descargas del navegador como se muestra en la Figura 5.20.

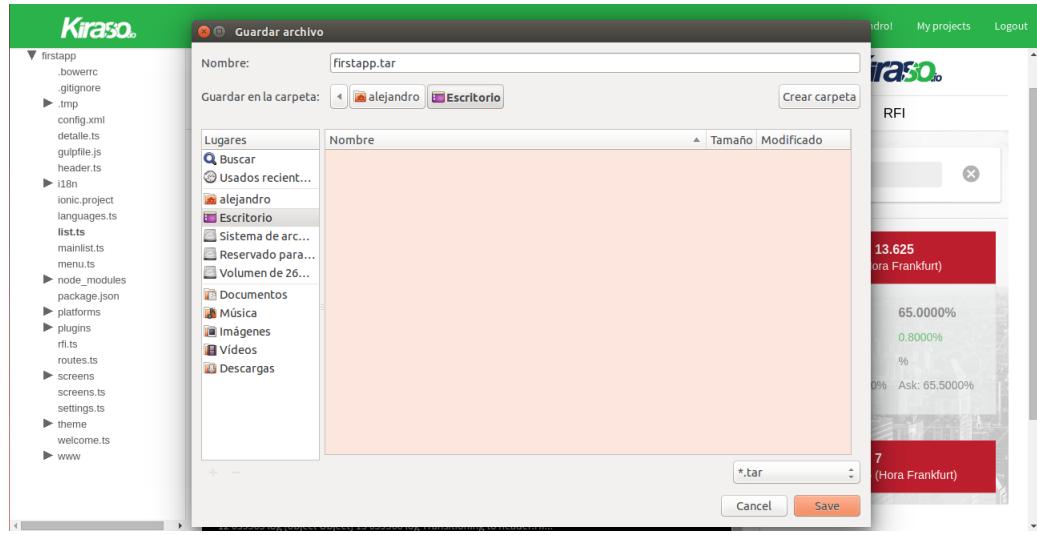


Figura 5.20: Descarga de archivo del Asistente para ensamblar aplicaciones móviles.

Finalmente, cada vez que el usuario requiere hacer un cambio de sección, decide salir de la aplicación o decide eliminar algún componente, siempre aparece una vista de verificación para que el usuario confirme su deseo de realizar dicha acción y prever un error humano. Un ejemplo de estas confirmaciones se muestra en la Figura 5.21.

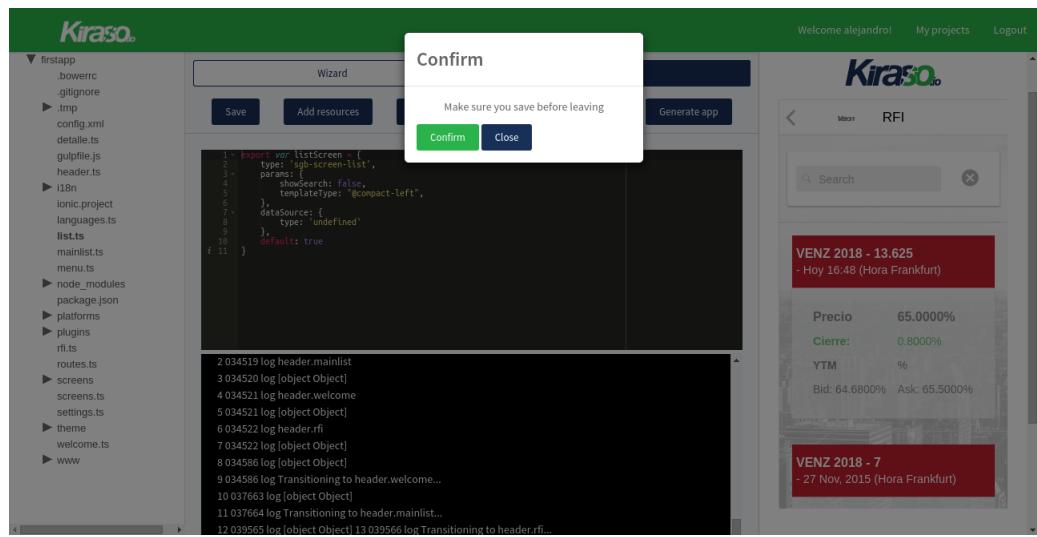


Figura 5.21: Ejemplo de confirmaciones del Asistente para ensamblar aplicaciones móviles.

## 5.4. Transición

La fase de transición no fue llevada a cabo debido a que el objetivo del proyecto desde su concepción fue la elaboración de un prototipo inicial. Es por esto que, el Asistente para ensamblar aplicaciones móviles aún no ha entrado en una fase formal de transición a producción.

# Capítulo 6

## Retos Enfrentados y Logros Adicionales

Este capítulo muestra aquellos retos que se presentaron a lo largo de la pasantía, así como la descripción de algunos logros adicionales que no estaban en el plan inicial.

### 6.1. Retos enfrentados

Durante el desarrollo del proyecto de pasantía surgieron diversos retos. El primero de ellos consistió en entender a plenitud el *framework* Kiraso. Este es un *framework* que si bien agrega un nivel de abstracción que facilita el desarrollo de aplicaciones móviles, a su vez tiene un cierto número de tecnologías involucradas que dificultó el entendimiento del mismo y de todas sus capacidades. Para cumplir el objetivo de esta pasantía se realizó un estudio profundo de su arquitectura y su funcionamiento, además de asistir a diversas reuniones con el cliente.

Adicionalmente, para el desarrollo de esta pasantía se incorporaron múltiples tecnologías que si bien se utilizaron para facilitar y agilizar el trabajo, en ciertos momentos trajo complicaciones debido a que, se generaban errores que eran difíciles de detectar o en otros casos, había que dedicar más tiempo del esperado para comprender su funcionamiento.

Otro reto importante de este desarrollo, que también se deriva del uso de múltiples tecnologías, fue mantener la modularidad y la integridad de todos los componentes de la aplicación. En ocasiones resultó complicado mantener la comunicación entre las distintas partes de la aplicación a fin de que la misma fuese consistente en todo momento con la información que

mostraba.

En menor medida y hacia el final del desarrollo, la inclusión de *websockets* y la funcionalidad para cargar archivos al servidor también representaron un reto, ya que su integración con el desarrollo necesitó de un trabajo de investigación y aprendizaje de nuevas tecnologías para que funcionaran debidamente.

## 6.2. Logros adicionales

Durante el desarrollo del proyecto de pasantía se alcanzaron una serie de logros adicionales que no se encontraban estipulados en el plan de trabajo inicial.

Entre ellos se encuentra la realización de la sección de vista previa del asistente y todas las operaciones que se pueden realizar en esta sección. La misma agrega un valor importante al asistente ya que permite al desarrollador la facilidad de trabajar todo desde un mismo lugar y a la vez, poder apreciar en tiempo real como sus cambios surten efecto en la aplicación móvil en desarrollo.

En esta sección de vista previa están presentes todas sus funcionalidades como son la vista y edición de los archivos fuentes, ejecución de la aplicación usando Kiraso, visualización a tiempo real de los procesos de Kiraso en ejecución en el servidor, vista previa de la aplicación móvil y agregar recursos (archivo comprimido) para personalizar aplicaciones. Además de esto se requirió de trabajo extra para mantener la conexión entre los cambios realizados en el editor de texto y la información que estaba almacenada en el componente gráfico del asistente.

Por otro lado, también se agregaron opciones como por ejemplo, al decidir que está terminada una aplicación, aparte de agregarla a la sección de aplicaciones precontruidas, también se genera un archivo .apk para ser instalado en plataformas Android y se da la opción de descargar el comprimido del proyecto completo junto con el .apk, el cual se encuentra en un subdirectorio del comprimido.

Si bien muchos de estos logros adicionales fueron decididos en conjunto con el cliente y se tomaron en cuenta para la planificación del desarrollo del proyecto, también es cierto que para cumplir el proyecto de manera satisfactoria y tomando en cuenta los aspectos adicionales se requirió de una cantidad de esfuerzo y tiempo representativa.

# Capítulo 7

## Conclusiones y Recomendaciones

En este proyecto de pasantía se desarrolló el Asistente para ensamblar aplicaciones móviles usando tecnología web que permite crear aplicaciones móviles multiplataforma usando el *framework* Kiraso de Synergy-GB. El proceso completo incluyó el diseño y desarrollo del *frontend* de la aplicación y su respectiva integración con el *backend* también diseñado y desarrollado durante este proceso.

Este Asistente para ensamblar aplicaciones permite crear aplicaciones móviles nuevas y estructurarlas gráficamente, configurar los distintos componentes, agregar, consultar y eliminar componentes, ver y descargar la estructura de archivos, editar los archivos de la aplicación, ejecutar la aplicación y ver su vista previa, ver los mensajes de los procesos a tiempo de ejecución, agregar recursos para personalizar las aplicaciones y una gestión básica de usuarios.

Esta solución permite a los usuarios o desarrolladores reducir los tiempos de desarrollo de aplicaciones móviles, además de ser una forma más amigable de familiarizarse con el *framework* Kiraso. Se espera que, una vez que este Asistente pase a producción dentro de la empresa, se convierta en una herramienta importante que le permita a Synergy-GB disminuir el esfuerzo requerido para desarrollar sus proyectos, aumentando su productividad.

Para el desarrollo del prototipo se utilizaron diversas herramientas entre las que destaca el *stack* de MEAN.js, es decir, MongoDB, Express, AngularJS y NodeJS. Así como los patrones de diseño: inyección de dependencias y observador. El aprendizaje de lo antes mencionado durante el proyecto de pasantía, resultó en una experiencia enriquecedora que permitió aumentar las destrezas del desarrollador.

Al finalizar el periodo establecido para el desarrollo de la pasantía se obtuvo un prototipo que cubre todos los objetivos planteados tanto por el cliente como por el plan de trabajo, así como también ciertas funcionalidades adicionales de interés para la empresa.

Este Asistente, que es una extensión del *framework* Kiraso, es una herramienta que ofrece ventajas tanto a Synergy-GB como a sus clientes, además de proveer una forma de trabajo que busca que los desarrolladores hagan productos de calidad y fáciles de mantener, por esto se recomienda altamente seguir desarrollando y trabajando en esta herramienta para que sus mejoras ofrezcan grandes beneficios a la empresa.

Dado que el desarrollo del Asistente para ensamblar aplicaciones móviles no ha sido finalizado en su totalidad, otra recomendación fundamental es la mejora de la gestión de usuarios. Es importante agregar características como seguridad y manejo de roles. Si el *framework* se va a comercializar junto con su Asistente es importante dar distintos niveles de usuarios, con distintos permisos para las acciones a realizar con el Asistente. Además de agregar un administrador que tenga control sobre los distintos usuarios registrados.

También se recomienda a la empresa que el desarrollo del *framework* vaya de la mano siempre con el desarrollo de este Asistente. Es decir, que a medida que se expanda y se agreguen funcionalidades o mejoras al *framework* Kiraso, se realice el desarrollo necesario, tanto en el *frontend* como en el *backend* del Asistente, para que de esta manera en todo momento abarque completamente las funcionalidades del *framework* y se le pueda sacar todo el provecho posible.

Por último, se recomienda que se realicen pruebas exhaustivas que garanticen la calidad del desarrollo.

Esta experiencia de pasantía le permitió al pasante buscar una solución a un problema real en donde tuvo que poner en práctica todos los conocimientos adquiridos a lo largo de la carrera universitaria. Por otro lado, se generó un crecimiento en cuanto a la gestión del tiempo y de cambios en los requerimientos, y estimación de tiempo de las actividades. Además, se dio la oportunidad de aprender nuevas herramientas y tecnologías, así como la oportunidad de trabajar en conjunto con el equipo de desarrollo de la empresa y participar en la toma de decisiones sobre cambios en el *framework* y agregar nuevas funcionalidades al mismo. Todo lo mencionado anteriormente, resultó muy beneficioso para el pasante ya que aumenta sus destrezas y habilidades en el ámbito laboral.

# Bibliografía

- [1] Synergy-GB, “Synergy-GB.” <http://synergy-gb.com>, consultado el 11 de marzo de 2016, 2013.
- [2] Synergy-GB, “Organigrama de Synergy-GB.” consultado el 11 de marzo de 2016, 2014.
- [3] Synergy-GB, “Áreas de la empresa.” consultado el 11 de marzo de 2016, 2014.
- [4] D. Riehle, “Framework design: A role modeling approach.” <http://dirkriehle.com/computer-science/research/dissertation/diss-a4.pdf>, consultado el 11 de marzo de 2016, 2000.
- [5] R. Guenther and J. Radebaugh, *Understanding Metadata*. NISO Press, 2004.
- [6] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web Services: Concepts, Architectures and Applications*. Springer Science, Business Media, 2004.
- [7] L. Richardson, *RESTFul Web Services for the Real World*. O'Reilly Media, Inc, 1 ed., 2007.
- [8] I. Evans, “Single page application (spa) una tendencia creciente.” <http://tecnimedios.com/blog/programacion/jquery/single-page-application-spa-una-tendencia-creciente/>, consultado el 11 de marzo de 2016, 2013.
- [9] I. Fette and A. Melnikov, “The websocket protocol.” <https://tools.ietf.org/html/rfc6455#section-1.2>, consultado el 11 de marzo de 2016, 2011.
- [10] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education, 1994.

- [11] R. F. Grove and E. Ozkan, “The mvc-web design pattern.” <http://grove.cs.jmu.edu/groverf/papers/WEBIST2011.pdf>, consultado el 14 de marzo de 2016, 2011.
- [12] M. Fowler, “Inversion of control containers and the dependency injection pattern.” <http://martinfowler.com/articles/injection.html>, consultado el 11 de marzo de 2016, 2004.
- [13] “Javascript.” <https://developer.mozilla.org/es/docs/Web/JavaScript>, consultado el 14 de marzo de 2016.
- [14] “TypeScript.” <https://github.com/Microsoft/TypeScript>, consultado el 14 de marzo de 2016.
- [15] “What is angular?.” <https://docs.angularjs.org/guide/introduction>, consultado el 14 de marzo de 2016.
- [16] “Node.js.” <https://nodejs.org/en/>, consultado el 14 de marzo de 2016.
- [17] “Express.” <http://expressjs.com/>, consultado el 14 de marzo de 2016.
- [18] “Introducción a express.js.” <https://geekytheory.com/introduccion-a-express-js/>, consultado el 14 de marzo de 2016.
- [19] “Reinventando la gestión de datos.” <https://www.mongodb.com/es>, consultado el 14 de marzo de 2016.
- [20] “Mongoose.” <http://mongoosejs.com/>, consultado el 14 de marzo de 2016.
- [21] “Socket.io.” <http://socket.io/>, consultado el 14 de marzo de 2016.
- [22] “D3 data-driven documents.” <https://d3js.org/>, consultado el 14 de marzo de 2016.
- [23] “Bower a package manager for the web.” <http://bower.io/>, consultado el 14 de marzo de 2016.
- [24] “What is npm?.” <https://docs.npmjs.com/getting-started/what-is-npm>, consultado el 14 de marzo de 2016.
- [25] “What is gulp?.” <https://github.com/gulpjs/gulp>, consultado el 14 de marzo de 2016.

- [26] “Jade.” <https://github.com/pugjs/jade>, consultado el 14 de marzo de 2016.
- [27] “Html introduction.” <https://developer.mozilla.org/en-US/docs/Web/HTML>, consultado el 14 de marzo de 2016.
- [28] “Sass (syntactically awesome stylesheets).” [http://sass-lang.com/documentation/file.SASS\\_REFERENCE.html](http://sass-lang.com/documentation/file.SASS_REFERENCE.html), consultado el 14 de marzo de 2016.
- [29] “Welcome to ionic.” <http://ionicframework.com/docs/guide/preface.html>, consultado el 14 de marzo de 2016.
- [30] “The web’s scaffolding tool for modern webapps.” <http://yeoman.io/>, consultado el 14 de marzo de 2016.
- [31] “Introducing JSON.” <http://www.json.org/>, consultado el 14 de marzo de 2016.
- [32] “Metodología de Poyectos Synergy-GB [Enfoque en Seguimiento].” consultado el 17 de enero de 2016.
- [33] K. Schwaber and J. Sutherland, “The Scrum Guide.” <https://www.scrum.org/scrum-guide>, consultado el 15 de enero de 2016, 2013.

# Glosario

## **Back-End**

Es la labor de ingeniería que compone el acceso a bases de datos, creación de servicios, y generación de plantillas del lado del servidor.

## **Front-End**

Es la maquetación de la estructura semántica del contenido (HTML), codificar el diseño en hojas de estilo (CSS) y agregar la interacción con el usuario (Javascript).

## **Stack**

Es un conjunto de subsistemas o componentes de software necesarios para crear una plataforma o solución tecnológica completa.

# **Apéndice A**

## **Lista de Riesgos**



**SYNERGY-GB**

## **Lista de Riesgos**

**Asistente para ensamblar  
aplicaciones móviles**

Asistente para ensamblar aplicaciones móviles basado en tecnología web.	Versión: 1.0
Lista de Riesgos	Fecha: 31/03/2016

## Historia de Revisión

Fecha	Versión	Descripción	Autor
31/03/2016	1.0	Lista de Riesgos	Alejandro Guillen

Asistente para ensamblar aplicaciones móviles basado en tecnología web.	Versión: 1.0
Lista de Riesgos	Fecha: 31/03/2016

## Tabla de Contenidos

1 Introducción .....	4
1.1 Propósito .....	4
1.2 Alcance .....	4
1.3 Referencias .....	4
1.4 Vista Global .....	4
2 Riesgos .....	5
2.1 Planificación Inadecuada .....	5
2.1.1 Magnitud del Riesgo: Alta .....	5
2.1.2 Descripción .....	5
2.1.3 Impactos .....	5
2.1.4 Indicadores .....	5
2.1.5 Estrategia de Mitigación .....	5
2.1.6 Plan de Contingencia.....	5
2.2 Requerimientos Cambiantes .....	6
2.2.1 Magnitud del Riesgo: Alta .....	6
2.2.2 Descripción .....	6
2.2.3 Impactos .....	6
2.2.4 Indicadores .....	6
2.2.5 Estrategia de Mitigación .....	6
2.2.6 Plan de Contingencia.....	6
2.3 Gestión Inadecuada de Riesgos.....	7
2.3.1 Magnitud del Riesgo: Alta .....	7
2.3.2 Descripción .....	7
2.3.3 Impactos .....	7
2.3.4 Indicadores .....	7
2.3.5 Estrategia de Mitigación .....	7
2.3.6 Plan de Contingencia.....	7

Asistente para ensamblar aplicaciones móviles basado en tecnología web.	Versión: 1.0
Lista de Riesgos	Fecha: 31/03/2016

# Lista de Riesgos

## 1. Introducción

En este documento se presenta una lista detallada de los riesgos que se considera pueden afectar - directa o indirectamente - el diseño y desarrollo exitoso del Asistente para ensamblar aplicaciones móviles basado en tecnología web.

### 1.1 Propósito

Este documento tiene como propósito servir de herramienta para evaluar los riesgos que se pudiesen presentar durante el desarrollo del proyecto y establecer un plan de detección y contención, para evitar que éstos (los riesgos) afecten el desarrollo y culminación del mismo.

### 1.2 Alcance

El alcance de este documento es analizar y examinar los riesgos que se podrían presentar en el proceso de diseño y desarrollo del prototipo a fin de lograr el propósito explicado en el apartado anterior.

### 1.3 Referencias

Para esta entrega del documento, no se hacen referencias a otros documentos.

### 1.4 Vista Global

En este documento, se presentará un listado de los riesgos en orden descendente de magnitud (gravedad e impacto). Para cada riesgo se detallarán su magnitud (indica el nivel de impacto en el prototipo o su desarrollo), descripción, impacto (en qué aspectos, y cómo afectará el riesgo), indicadores (describe como monitorear y detectar que el riesgo está a punto de ocurrir), estrategia de mitigación (describe qué se está haciendo actualmente para evitar el riesgo), y plan de contingencia (describe cuál será el procedimiento a seguir en caso de que el riesgo se materialice).

Asistente para ensamblar aplicaciones móviles basado en tecnología web.	Versión: 1.0
Lista de Riesgos	Fecha: 31/03/2016

## 2. Riesgos

### 2.1 Planificación Inadecuada

#### 2.1.1 *Magnitud del Riesgo: Alta*

#### 2.1.2 *Descripción*

La planificación inadecuada se refiere a la elaboración de un cronograma de actividades que en principio parece factible, pero que luego en la práctica se hace imposible de seguir. Este riesgo se asocia a una incorrecta estimación del esfuerzo requerido para realizar actividades claves para el desarrollo satisfactorio del proyecto.

#### 2.1.3 *Impactos*

- Incumplimiento de las actividades y los objetivos planteados.
- Acumulación del trabajo y agotamiento del desarrollador para cumplir con los objetivos trazados.
- Incumplimiento del objetivo principal del proyecto, pudiendo ocasionar que la aplicación no cumpla con las expectativas del cliente.

#### 2.1.4 *Indicadores*

- Evidencias de fallas o retrasos en las primeras entregas tanto de documentación como de código, podrían indicar que la planificación no ha sido elaborada correctamente.

#### 2.1.5 *Estrategia de Mitigación*

Para reducir la probabilidad de que este riesgo ocurra, se cuenta con un plan donde se establecen los objetivos de cada fase y algunas actividades que deben llevarse. Además se propone:

- Realizar seguimiento desde el inicio hasta la culminación de cada actividad.
- Planificar tiempo suficiente para que el desarrollador se familiarice con las herramientas a utilizar antes de iniciar el desarrollo del proyecto.

#### 2.1.6 *Plan de Contingencia*

Convocar a una reunión de los miembros del equipo lo más pronto posible, en la cual se evaluarían las razones del retraso y las posibles soluciones para mejorar la situación, solventar los inconvenientes causados y realizar un nuevo plan de acción.

Asistente para ensamblar aplicaciones móviles basado en tecnología web.	Versión: 1.0
Lista de Riesgos	Fecha: 31/03/2016

## 2.2 Requerimientos Cambiantes

### 2.2.1 *Magnitud del Riesgo: Alta*

#### 2.2.2 *Descripción*

Este riesgo está asociado a la aparición de nuevos requerimientos o al cambio de alguno ya establecido, lo cual podría perjudicar la planificación del desarrollo del proyecto.

#### 2.2.3 *Impactos*

- Aumento del esfuerzo por parte del desarrollador.
- Incumplimiento de las actividades planificadas inicialmente.
- El prototipo final podría no cumplir con las expectativas del cliente.

#### 2.2.4 *Indicadores*

- Solicitud del cliente de incluir nuevas funcionalidades.
- En medio del desarrollo aparecen restricciones que no fueron contempladas al momento de especificar los requerimientos.
- Solicitud expresa del cliente de cambiar uno o varias de las funcionalidades ya desarrolladas.

#### 2.2.5 *Estrategia de Mitigación*

- Realizar un buen levantamiento inicial de información y requerimientos.
- Realizar reuniones periódicas con el cliente a fin de ir validando las funcionalidades que se van desarrollando.

#### 2.2.6 *Plan de Contingencia*

Realizar un estudio para determinar el impacto que tendrían los cambios en el cronograma establecido, el esfuerzo requerido para realizar los cambios y establecer el nuevo cronograma del proyecto junto con el alcance del mismo.

.

Asistente para ensamblar aplicaciones móviles basado en tecnología web. Lista de Riesgos	Versión: 1.0 Fecha: 31/03/2016
---	-----------------------------------

## 2.3 Gestión Inadecuada de Riesgos

### 2.3.1 *Magnitud del Riesgo: Alta*

#### 2.3.2 *Descripción*

Está asociado a la mala gestión de los riesgos planteados en este documento o a la desestimación de los indicadores de que uno o más riesgos están a punto de materializarse.

#### 2.3.3 *Impactos*

- Que se concrete uno o más de los riesgos desestimados.
- Incumplimiento de los objetivos planteados.
- El prototipo final no cumple con las funcionalidades especificadas

#### 2.3.4 *Indicadores*

- Se está ignorando la aparición de uno o varios indicadores de riesgos.
- No se está llevando a cabo las estrategias de mitigación de los demás riesgos.

#### 2.3.5 *Estrategia de Mitigación*

- Tener en cuenta, durante todo el desarrollo del prototipo, los riesgos descritos en este documentos.
- Realizar las consideraciones pertinentes y tomar las medidas necesarias para evitar que los riesgos aquí planteados sean mal gestionados.
- Llevar a cabo todas las estrategias de mitigación aquí descritas.
- Realizar una revisión periódica para asegurar que los riesgos están siendo mitigados.

#### 2.3.6 *Plan de Contingencia*

En caso de concretarse este riesgo, y que aunado a ello se materialicen otros riesgos mencionados en el documento, se proseguirá a atacar cada riesgo mediante la ejecución de su plan de contingencia correspondiente.

## **Apéndice B**

### **Casos de Uso**



**SYNERGY-GB**

## Casos de Uso

### **Asistente para ensamblar aplicaciones móviles**

## **Tabla de Contenido**

Introducción .....	2
Descripción de los Casos de Uso .....	2
Tabla de Actores/Interesados .....	3
Tabla de Casos de Uso .....	4
Diagrama de Casos de Uso.....	5
Especificaciones de los Casos de Uso .....	6
Aspectos sobre la Organización .....	14
Aspectos Legales .....	14

## Introducción

El presente documento contiene las especificaciones de los casos de uso definidos para cumplir con los requerimientos de software del Asistente para ensamblar aplicaciones móviles.

## Descripción de los Casos de Uso

En esta sección describen los casos de uso del sistema.

Número	Nombre del Caso de Uso	Descripción
AK_1	Crear usuario	Permite a un usuario registrarse en la aplicación.
AK_2	Iniciar sesión	Permite a una persona registrada ingresar a la aplicación indicando usuario y contraseña.
AK_3	Cerrar sesión	Permite a un usuario salir de la aplicación.
AK_4	Listar aplicación	Permite al usuario ver sus aplicaciones.
AK_5	Crear aplicación	Permite a un usuario crear una aplicación nueva.
AK_6	Gestionar aplicación	Permite a un usuario consultar sus aplicaciones y editarlas.
AK_7	Guardar aplicación	Permite a un usuario guardar una aplicación en desarrollo.
AK_8	Eliminar aplicación	Permite al usuario eliminar una de sus aplicaciones.
AK_9	Ejecutar aplicación	Permite al usuario ejecutar una aplicación en desarrollo usando Kiraso.
AK_10	Clonar aplicación	Permite al usuario utilizar una aplicación existente y finalizada para editarla.
AK_11	Mostrar vista previa	Permite mostrar la vista previa de la aplicación en desarrollo en ejecución.
AK_12	Agregar componente	Permite a un usuario agregar un nodo como componente al grafo de su aplicación en desarrollo.
AK_13	Eliminar un componente	Permite a un usuario eliminar un componente.
AK_14	Configurar componente	Permite a un usuario configurar un componente para su aplicación.
AK_15	Consultar componente	Permite a un usuario consultar la documentación de un componente.

AK_16	Clonar componente	Permite a un usuario clonar un componente desde un repositorio Github.
AK_17	Gestionar archivos	Permite a un usuario ver y editar los archivos de la aplicación en desarrollo.
AK_18	Ver archivo	Permite a un usuario consultar un archivo desde un asistente.
AK_19	Editar archivo	Permite a un usuario editar un archivo desde el asistente.
AK_20	Guardar archivo	Permite a un usuario guardar un archivo editado desde el asistente.
AK_21	Agregar recursos	Permite a un usuario agregar recursos necesarios para la personalización de aplicaciones.
AK_22	Descargar estructura de archivos	Permite a un usuario descargar desde el asistente la estructura de archivos del proyecto Kiraso en desarrollo.
AK_23	Listar estructura de archivos	Permite a un usuario visualizar la estructura de archivos del proyecto Kiraso en desarrollo.

### Tabla de Actores/Interesados

En esta sección se describen los stakeholders.

Nombre	Descripción	Responsabilidades
Pasante de Proyecto	Es la persona designada por la empresa para realizar el diseño y desarrollo de la aplicación.	Diseñar e implementar la aplicación.
Líder de Investigación y desarrollo	Es quien supervisa el desarrollo del proyecto, propone y aprueba decisiones acerca de la arquitectura del sistema.	Supervisar el desarrollo del proyecto. Verificar que la planificación se cumpla. Aprobar decisiones de alto nivel acerca de la arquitectura del sistema.
Cliente	Es la persona encargada de definir los requerimientos según los cuales se desarrollara la aplicación.	Contribuir al levantamiento de requerimientos y la confirmación de los mismos durante todo el desarrollo.
Usuario o desarrollador	Es el usuario final de la aplicación.	Ensamblar y modificar aplicaciones móviles.

## Tabla de Casos de Uso

A continuación se presenta la tabla de casos de uso:

Número	Nombre del Caso de Uso	Actor
AK_1	Crear usuario	Usuario (Desarrollador)
AK_2	Iniciar sesión	Usuario (Desarrollador)
AK_3	Cerrar sesión	Usuario (Desarrollador)
AK_4	Listar aplicaciones	Usuario (Desarrollador)
AK_5	Crear aplicación	Usuario (Desarrollador)
AK_6	Gestionar aplicación	Usuario (Desarrollador)
AK_7	Guardar aplicación	Usuario (Desarrollador)
AK_8	Eliminar aplicación	Usuario (Desarrollador)
AK_9	Ejecutar aplicación	Usuario (Desarrollador)
AK_10	Clonar aplicación	Usuario (Desarrollador)
AK_11	Mostrar vista previa	Usuario (Desarrollador)
AK_12	Agregar componente	Usuario (Desarrollador)
AK_13	Eliminar un componente	Usuario (Desarrollador)
AK_14	Configurar componente	Usuario (Desarrollador)
AK_15	Consultar componente	Usuario (Desarrollador)
AK_16	Clonar componente	Usuario (Desarrollador)
AK_17	Gestionar archivos	Usuario (Desarrollador)
AK_18	Ver archivo	Usuario (Desarrollador)
AK_19	Editar archivo	Usuario (Desarrollador)
AK_20	Guardar archivo	Usuario (Desarrollador)
AK_21	Agregar recursos	Usuario (Desarrollador)
AK_22	Descargar estructura de archivos	Usuario (Desarrollador)
AK_23	Listar estructura de archivos	Usuario (Desarrollador)

## Diagrama de Casos de Uso

A continuación se presenta el Diagrama de los Casos de uso del sistema:



## Especificaciones de los Casos de Uso

<b>AK_1: Crear usuario</b>	
Descripción: Permite a un usuario registrarse en la aplicación.	
Precondición: Los servicios del asistente deben estar disponibles.	
FLUJO BASICO:	
ACTOR	SISTEMA
1 El usuario ingresa en la página de registro de usuario.	2 Muestra el formulario de registro de usuario.
3 El desarrollador introduce su usuario, contraseña y confirmación de contraseña.	4 Registra al usuario exitosamente y redirecciona a la página de inicio de sesión.
FLUJOS ALTERNOS:	
ACTOR	SISTEMA
	4.1 Detecta si las contraseñas no coinciden, si el usuario deja alguno de los campos vacíos o si el usuario ya existe.
	4.2 Señala errores al usuario y vuelve al paso 2
Post condición: El usuario queda registrado en la aplicación.	
Requerimientos especiales: Ninguno	
Puntos de extensión: Ninguno	

<b>AK_2: Iniciar sesión</b>	
Descripción: Permite a una persona registrada ingresar a la aplicación indicando usuario y contraseña.	
Precondición: Los servicios del asistente deben estar disponibles.	
FLUJO BASICO:	
ACTOR	SISTEMA
1 El usuario ingresa en la página principal de la aplicación.	2 Muestra el formulario de inicio de sesión.
3 El usuario introduce su usuario y contraseña.	4 Autentica al usuario exitosamente y redirecciona a la vista de la lista de aplicaciones del usuario.
FLUJOS ALTERNOS:	
ACTOR	SISTEMA
	4.1 Detecta si el usuario deja alguno de los campos vacíos o si el usuario y/o contraseñas son equivocados.
	4.2 Señala los errores al usuario y vuelve al paso 2.

Post condición: Se inicia sesión del usuario luego de ser autenticado.
Requerimientos especiales: Ninguno
Puntos de extensión: Ninguno

### AK\_3: Cerrar Sesión

Descripción: Permite a un usuario cerrar la sesión cuando lo desee.

Precondición: Ninguna.

#### FLUJO BASICO:

ACTOR	SISTEMA
1 El usuario selecciona la opción Cerrar Sesión en la barra de navegación.	2 Re-direcciona al usuario a la página de login.

FLUJOS ALTERNOS: Ninguno.

Post condición: La aplicación cerró la sesión del usuario.

Requerimientos especiales: Ninguno

Puntos de extensión: Ninguno

### AK\_4: Listar aplicaciones

Descripción: Permite al usuario ver sus aplicaciones.

Precondición: Los servicios del asistente deben estar disponibles.

#### FLUJO BASICO:

ACTOR	SISTEMA
1 El usuario inicia sesión exitosamente.	2 Muestra la lista de aplicaciones.

FLUJOS ALTERNOS:

ACTOR	SISTEMA
1.1 El usuario hace clic en la barra de navegación en la sección de mis proyectos.	2.1 Se solicita confirmación para dejar la vista del asistente.
1.2.1 El usuario confirma	2.2 Va al paso 2
1.2.2 El usuario cancela la operación.	

Post condición: Se presenta al usuario la lista de aplicaciones.

Requerimientos especiales: Ninguno

Puntos de extensión: Ninguno

### AK\_5: Crear aplicación

Descripción: Permite a un usuario crear una aplicación nueva.

Precondición: Los servicios del asistente deben estar disponibles.

#### FLUJO BASICO:

ACTOR	SISTEMA
1 El usuario hace clic en el botón de proyecto nuevo.	2 Muestra el formulario solicitando el nombre de la nueva aplicación.
3 El usuario ingresa el nombre de la nueva	4 El usuario es re-direccionado al asistente para

aplicación	ensamblar aplicaciones móviles.
<b>FLUJOS ALTERNOS:</b>	
ACTOR	SISTEMA
	4.1 Señala los errores al usuario y vuelve al paso 2.
Post condición: Una nueva aplicación es creada.	
Requerimientos especiales: Ninguno	
Puntos de extensión: Ninguno	

#### **AK\_6: Gestionar aplicación**

Descripción: Permite a un usuario consultar sus aplicaciones y editarlas.

Precondición: Los servicios del asistente deben estar disponibles.

#### **FLUJO BASICO:**

ACTOR	SISTEMA
1 El usuario selecciona una aplicación existente de su lista de proyectos	2 Re-direcciona a la pantalla de ensamblaje de aplicaciones.

#### **FLUJOS ALTERNOS:** Ninguno

Post condición: Se seleccionó una aplicación.

Requerimientos especiales: Ninguno

Puntos de extensión: Guardar aplicación, Eliminar aplicación, Ejecutar aplicación, Clonar aplicación, Mostrar vista previa, Agregar componente, Eliminar componente, Configurar componente, Consultar componente, Clonar componente, Gestionar archivos.

#### **AK\_7: Guardar aplicación**

Descripción: Permite a un usuario guardar una aplicación en desarrollo.

Precondición: Los servicios del asistente deben estar disponibles.

#### **FLUJO BASICO:**

ACTOR	SISTEMA
1 El usuario selecciona la opción de guardar en la vista gráfica del asistente.	2 Muestra mensaje de la operación.

#### **FLUJOS ALTERNOS:** Ninguno

Post condición: La aplicación fue guardada exitosamente.

Requerimientos especiales: Ninguno

Puntos de extensión: Ninguno

#### **AK\_8: Eliminar aplicación**

Descripción: Permite a un usuario eliminar una de sus aplicaciones.

Precondición: Los servicios del asistente deben estar disponibles.

#### **FLUJO BASICO:**

ACTOR	SISTEMA
-------	---------

1 El usuario selecciona la opción de eliminar aplicación de la lista de aplicaciones.	2 Muestra ventana de confirmación.
3 El usuario confirma que desea eliminar el elemento.	4 Elimina el elemento.
<b>FLUJOS ALTERNOS:</b>	
ACTOR	SISTEMA
3.1 El usuario cancela la eliminación del elemento.	
Post condición: Se elimina una aplicación.	
Requerimientos especiales: Ninguno	
Puntos de extensión: Ninguno	

#### **AK\_9: Ejecutar aplicación**

Descripción: Permite al usuario ejecutar una aplicación en desarrollo usando Kiraso.

Precondición: Los servicios del asistente deben estar disponibles.

**FLUJO BASICO:**

ACTOR	SISTEMA
1 El usuario selecciona la opción de ejecutar aplicación en la sección de vista previa.	2 Ejecuta la aplicación y muestra los procesos a tiempo real.
<b>FLUJOS ALTERNOS:</b> Ninguno	
Post condición: Se ejecuta la aplicación.	
Requerimientos especiales: Ninguno	
Puntos de extensión: Ninguno	

#### **AK\_10: Clonar aplicación**

Descripción: Permite al usuario utilizar una aplicación existente y finalizada para editarla.

Precondición: Los servicios del asistente deben estar disponibles.

**FLUJO BASICO:**

ACTOR	SISTEMA
1 El usuario selecciona una aplicación en el panel de aplicaciones pre-construidas.	2 Muestra ventana de confirmación.
3 El usuario confirma	4 Carga el grafo de la aplicación seleccionada.
<b>FLUJOS ALTERNOS:</b> Ninguno.	
ACTOR	
3.1 El usuario cancela	
Post condición: Se carga el grafo de una aplicación existente.	
Requerimientos especiales: Ninguno	
Puntos de extensión: Ninguno	

#### **AK\_11: Mostrar vista previa**

<p><b>Descripción:</b> Permite mostrar la vista previa de la aplicación en desarrollo en ejecución.</p> <p><b>Precondición:</b> Los servicios del asistente deben estar disponibles.</p>	
<b>FLUJO BASICO:</b>	
ACTOR	SISTEMA
1 El usuario selecciona la opción de mostrar vista previa en la sección de vista previa.	2 Muestra la vista previa de la aplicación en ejecución.
<b>FLUJOS ALTERNOS:</b> Ninguno	
<b>Post condición:</b> Se muestra la vista previa de la aplicación en ejecución.	
<b>Requerimientos especiales:</b> Ninguno	
<b>Puntos de extensión:</b> Ninguno	

<p><b>AK_12: Agregar componente</b></p> <p><b>Descripción:</b> Permite a un usuario agregar un nodo como componente al grafo de su aplicación en desarrollo.</p> <p><b>Precondición:</b> Los servicios del asistente deben estar disponibles.</p>	
<b>FLUJO BASICO:</b>	
ACTOR	SISTEMA
1 El usuario selecciona un componente en el panel de componentes.	2 Agrega el componente al grafo.
<b>FLUJOS ALTERNOS:</b> Ninguno	
<b>Post condición:</b> Se agrega un componente al grafo.	
<b>Requerimientos especiales:</b> Ninguno	
<b>Puntos de extensión:</b> Ninguno	

<p><b>AK_13: Eliminar componente</b></p> <p><b>Descripción:</b> Permite a un usuario eliminar un componente.</p> <p><b>Precondición:</b> Los servicios del asistente deben estar disponibles.</p>	
<b>FLUJO BASICO:</b>	
ACTOR	SISTEMA
1 El usuario selecciona la opción de eliminar componente.	2 Muestra ventana de confirmación.
3 El usuario confirma.	4 Se elimina el componente.
<b>FLUJOS ALTERNOS:</b>	
ACTOR	SISTEMA
3.1 El usuario cancela.	
<b>Post condición:</b> Se elimina el componente.	
<b>Requerimientos especiales:</b> Ninguno	
<b>Puntos de extensión:</b> Ninguno	

<b>AK_14: Configurar componente</b>
-------------------------------------

Descripción: Permite a un usuario configurar un componente para su aplicación.	
Precondición: Los servicios del asistente deben estar disponibles.	
<b>FLUJO BASICO:</b>	
ACTOR	SISTEMA
1 El usuario selecciona un componente de su grafo.	2 Muestra los formularios de configuración en el panel dinámico.
3 El usuario completa los formularios con la información requerida.	
4 Guarda la información proporcionada	5 Almacena la información en los nodos correspondientes.
<b>FLUJOS ALTERNOS:</b> Ninguno	
Post condición: Se configuraron los componentes de la aplicación.	
Requerimientos especiales: Ninguno	
Puntos de extensión: Ninguno	

<b>AK_15: Consultar componente</b>	
Descripción: Permite a un usuario consultar la documentación de un componente.	
Precondición: Los servicios del asistente deben estar disponibles.	
<b>FLUJO BASICO:</b>	
ACTOR	SISTEMA
1 El usuario selecciona la opción de consultar documentación.	2 Re-direcciona al repositorio Github con la documentación del componente.
<b>FLUJOS ALTERNOS:</b> Ninguno	
Post condición: Se muestra la documentación del componente.	
Requerimientos especiales: Ninguno	
Puntos de extensión: Ninguno	

<b>AK_16: Clonar componente</b>	
Descripción: Permite a un usuario clonar un componente desde un repositorio Github.	
Precondición: Los servicios del asistente deben estar disponibles.	
<b>FLUJO BASICO:</b>	
ACTOR	SISTEMA
1 El usuario selecciona la opción de agregar un componente	2 Se muestra el formulario para agregar un nuevo componente.
3 El usuario introduce los datos necesarios.	4 Clona el componente dado.
<b>FLUJOS ALTERNOS:</b>	
ACTOR	SISTEMA
	4.1 Detecta si alguno de los campos está vacío y se lo indica al usuario.
	4.2 Señala errores al usuario y vuelve al paso 2.

Post condición: Se clona un componente.

Requerimientos especiales: Ninguno

Puntos de extensión: Ninguno

#### **AK\_17: Gestionar archivos**

Descripción: Permite a un usuario ver y editar los archivos de la aplicación en desarrollo.

Precondición: Los servicios del asistente deben estar disponibles.

##### **FLUJO BASICO:**

ACTOR	SISTEMA
1 El usuario selecciona la sección de vista previa.	2 Muestra ventana de confirmación
3 El usuario confirma.	4 Re-direcciona a la sección de vista previa.

##### **FLUJOS ALTERNOS:**

ACTOR	SISTEMA
3.1 El usuario cancela	

Post condición: Se pueden gestionar los archivos.

Requerimientos especiales: Ninguno

Puntos de extensión: Ver archivo, Editar archivo, Guardar archivo, Agregar recursos, Descargar estructura de archivos, Listar estructura de archivos.

#### **AK\_18: Ver archivo**

Descripción: Permite a un usuario consultar un archivo desde el asistente.

Precondición: Los servicios del asistente deben estar disponibles.

##### **FLUJO BASICO:**

ACTOR	SISTEMA
1 El usuario selecciona un archivo.	2 Muestra el contenido del archivo en el editor web.

##### **FLUJOS ALTERNOS: Ninguno**

Post condición: Se muestra el contenido de un archivo.

Requerimientos especiales: Ninguno

Puntos de extensión: Ninguno

#### **AK\_19: Editar archivo**

Descripción: Permite a un usuario editar un archivo desde el asistente.

Precondición: Los servicios del asistente deben estar disponibles.

##### **FLUJO BASICO:**

ACTOR	SISTEMA
1 El usuario modifica el contenido de un archivo en el editor.	2 Muestra el archivo con las modificaciones.

##### **FLUJOS ALTERNOS: Ninguna**

Post condición: Se edita el contenido de un archivo

Requerimientos especiales: Ninguno

Puntos de extensión: Ninguno

#### **AK\_20: Guardar archivo**

Descripción: Permite a un usuario guardar un archivo editado desde el asistente.

Precondición: Ninguna.

#### **FLUJO BASICO:**

<b>ACTOR</b>	<b>SISTEMA</b>
--------------	----------------

1 El usuario selecciona la opción de guardar archivo.	2 Guarda exitosamente el archivo.
---	-----------------------------------

#### **FLUJOS ALTERNOS: Ninguno**

Post condición: Se guarda un archivo exitosamente

Requerimientos especiales: Ninguno

Puntos de extensión: Ninguno

#### **AK\_21: Agregar recursos**

Descripción: Permite a un usuario agregar recursos necesarios para la personalización de aplicaciones.

Precondición: Los servicios del asistente deben estar disponibles.

#### **FLUJO BASICO:**

<b>ACTOR</b>	<b>SISTEMA</b>
--------------	----------------

1 El usuario selecciona la opción de agregar recursos.	2 Muestra el formulario para seleccionar un archivo
--	---

3 El usuario selecciona el archivo.	4 Muestra el archivo seleccionado
-------------------------------------	-----------------------------------

5 El usuario selecciona agregar recursos	6 Carga el archivo de recursos en el directorio <i>theme</i>
--	--

#### **FLUJOS ALTERNOS:**

<b>ACTOR</b>	<b>SISTEMA</b>
--------------	----------------

5.1 El usuario cancela la operación	
-------------------------------------	--

	6.1 Muestra mensaje de error si el formato del archivo no es el adecuado. Vuelve al paso 2.
--	---

Post condición: Se agregan recursos a la aplicación.

Requerimientos especiales: Ninguno

Puntos de extensión: Ninguno

#### **AK\_22: Descargar estructura de archivos**

Descripción: Permite a un usuario descargar desde el asistente la estructura de archivos del proyecto Kiraso en desarrollo.

Precondición: Los servicios del asistente deben estar disponibles.	
<b>FLUJO BASICO:</b>	
ACTOR	SISTEMA
1 El usuario selecciona la opción descarga	2 Muestra ventana para guardar archivo.
3 El usuario escoge nombre para guardar la aplicación.	4 Descarga el comprimido con los archivos.
<b>FLUJOS ALTERNOS:</b>	
ACTOR	SISTEMA
3.1 El usuario cancela la descarga.	
Post condición: Se descarga la estructura de archivos.	
Requerimientos especiales: Ninguno	
Puntos de extensión: Ninguno	

<b>AK_23: Listar estructura de archivos</b>	
Descripción: Permite a un usuario visualizar la estructura de archivos del proyecto Kiras en desarrollo.	
Precondición: Los servicios del asistente deben estar disponibles.	
<b>FLUJO BASICO:</b>	
ACTOR	SISTEMA
1 El usuario selecciona la opción de ir a la sección de vista previa.	2 Se construye la aplicación y se muestra la lista de archivos de la aplicación.
<b>FLUJOS ALTERNOS:</b> Ninguno	
Post condición: Se muestra la lista de archivos de una aplicación.	
Requerimientos especiales: Ninguno	
Puntos de extensión: Ninguno	

## Aspectos sobre la Organización

No aplica.

## Aspectos Legales

- Synergy-GB será dueña de todo lo que desarrolle durante la pasantía.
- No se podrán divulgar los detalles específicos de la implementación, el código, los datos de las bases de datos, la configuración de servidores ni cualquier otra información sensible que pueda ser utilizada por la competencia de Synergy-GB o agentes maliciosos para perjudicar al producto desarrollado durante la pasantía o la empresa.
- En el libro de pasantía se podrá utilizar el diseño de la solución, diagramas generales, de clases, arquitecturas y todo lo necesario para presentar y argumentar el proyecto ante la universidad.

## **Apéndice C**

# **Documento de Arquitectura de Software**



**SYNERGY-GB**

**Documento de Arquitectura del  
Software**

**Asistente para ensamblar  
aplicaciones móviles**

## Tabla de contenido

Introducción .....	3
Representación Arquitectónica .....	4
Vista de Escenarios .....	5
Vista Lógica .....	7
Vista de Procesos .....	7
Vista de Desarrollo .....	7
Vista de Despliegue o Implantación .....	8
Tamaño y Desempeño .....	9
Calidad .....	9
Referencias .....	11

## Introducción

Este documento provee una visión general de la arquitectura del Asistente para ensamblar aplicaciones móviles basado en tecnología web, usando distintas vistas arquitectónicas para representar distintos aspectos del sistema, brindando al lector una visión global y comprensible del diseño general del mismo.

El modelado de las vistas permitirá crear un modelo del sistema mucho más completo y constituye una base sólida para su desarrollo. En este documento se incluyen la vista de escenarios representada en el Diagrama de Casos de Uso, la vista de desarrollo ilustrada por el Diagrama de componentes y la vista de despliegue representada por el Diagrama de Despliegue.

## Representación Arquitectónica

La Arquitectura de Software básicamente es considerada como una vista del sistema donde se incluyen sus componentes principales y la interacción y coordinación entre ellos, necesarios para cumplir con los requerimientos específicos del sistema.

En este documento se utiliza el Modelo 4+1 Vistas de Kruchten, el cual organiza la descripción de la arquitectura de software en cinco vistas (Kruchten, 1995), donde se exponen las decisiones de diseño y una quinta vista para ilustrar y validar dichas decisiones. Estas vistas se componen por la de escenarios, lógica, procesos, desarrollo e implantación. Es importante destacar que éstas son complementadas con elementos de UML que proporcionen más información a nivel gráfico. Las vistas son las siguientes:

- La vista de escenarios, también llamada casos de uso, contiene los casos de uso críticos del sistema, representados mediante el diagrama de casos de uso. Es de gran importancia ya que a través de esta vista se puede entender mejor la funcionalidad del sistema y funciona como un indicador que ayuda al diseñador a descubrir los elementos de la arquitectura durante su diseño (Kruchten, 1995).
- La vista lógica comprende las abstracciones fundamentales del sistema a partir del dominio del problema, trata principalmente los requerimientos funcionales del sistema y forma la estructura del mismo, incluye el diagrama de clases (Kruchten, 1995).
- La vista de procesos describe el diseño de concurrencia y aspectos de sincronización y muestra algunos de los requisitos no funcionales, como son desempeño, rendimiento, seguridad, fiabilidad, entre otros (Kruchten, 1995).
- La vista de desarrollo o implantación expone las características y requisitos que debe cumplir el sistema a nivel técnico (Kruchten, 1995).
- La vista de implementación engloba la distribución de paquetes de la aplicación y de manera general la distribución y funcionamiento de las distintas capas del sistema. Describe el mapeo del software en el hardware (Kruchten, 1995).

## Vista de Escenarios

Para ilustrar la vista de escenarios, se incluye el diagrama de casos de uso definitivo.



Figura 1. Diagrama de Casos de Uso definitivo.

A continuación se presenta, la Tabla 1, donde se describen los casos de uso del sistema.

Número	Nombre del Caso de Uso	Descripción
AK_1	Crear usuario	Permite a un usuario registrarse en la aplicación.
AK_2	Iniciar sesión	Permite a una persona registrada ingresar a la aplicación indicando usuario y contraseña.
AK_3	Cerrar sesión	Permite a un usuario salir de la aplicación.
AK_4	Listar aplicación	Permite al usuario ver sus aplicaciones.
AK_5	Crear aplicación	Permite a un usuario crear una aplicación nueva.
AK_6	Gestionar aplicación	Permite a un usuario consultar sus aplicaciones y editarlas.
AK_7	Guardar aplicación	Permite a un usuario guardar una aplicación en desarrollo.
AK_8	Eliminar aplicación	Permite al usuario eliminar una de sus aplicaciones.
AK_9	Ejecutar aplicación	Permite al usuario ejecutar una aplicación en desarrollo usando Kiraso.
AK_10	Clonar aplicación	Permite al usuario utilizar una aplicación existente y finalizada para editarla.
AK_11	Mostrar vista previa	Permite mostrar la vista previa de la aplicación en desarrollo en ejecución.
AK_12	Agregar componente	Permite a un usuario agregar un nodo como componente al grafo de su aplicación en desarrollo.
AK_13	Eliminar un componente	Permite a un usuario eliminar un componente.
AK_14	Configurar componente	Permite a un usuario configurar un componente para su aplicación.
AK_15	Consultar componente	Permite a un usuario consultar la documentación de un componente.
AK_16	Clonar componente	Permite a un usuario clonar un componente desde un repositorio Github.
AK_17	Gestionar archivos	Permite a un usuario ver y editar los archivos de la aplicación en desarrollo.
AK_18	Ver archivo	Permite a un usuario consultar un archivo desde un asistente.

AK_19	Editar archivo	Permite a un usuario editar un archivo desde el asistente.
AK_20	Guardar archivo	Permite a un usuario guardar un archivo editado desde el asistente.
AK_21	Agregar recursos	Permite a un usuario agregar recursos necesarios para la personalización de aplicaciones.
AK_22	Descargar estructura de archivos	Permite a un usuario descargar desde el asistente la estructura de archivos del proyecto Kiraso en desarrollo.
AK_23	Listar estructura de archivos	Permite a un usuario visualizar la estructura de archivos del proyecto Kiraso en desarrollo.

Tabla 1. Descripción de Casos de Uso.

## Vista Lógica

La vista lógica no fue desarrollada, dado que la aplicación no sigue un estilo orientado a objetos.

## Vista de Procesos

La Vista de Procesos no fue desarrollada, dado que la aplicación no resuelve aspectos de concurrencia de usuarios y de datos.

## Vista de Desarrollo

La vista de desarrollo se realizó a través de un diagrama de componentes, donde se muestra la relación entre los mismos. Cabe destacar que solo los componentes más importantes están presentes en el diagrama de modo que se facilite su comprensión.

A continuación, en la figura 4, se muestra el diagrama de componentes (resumido) del sistema.

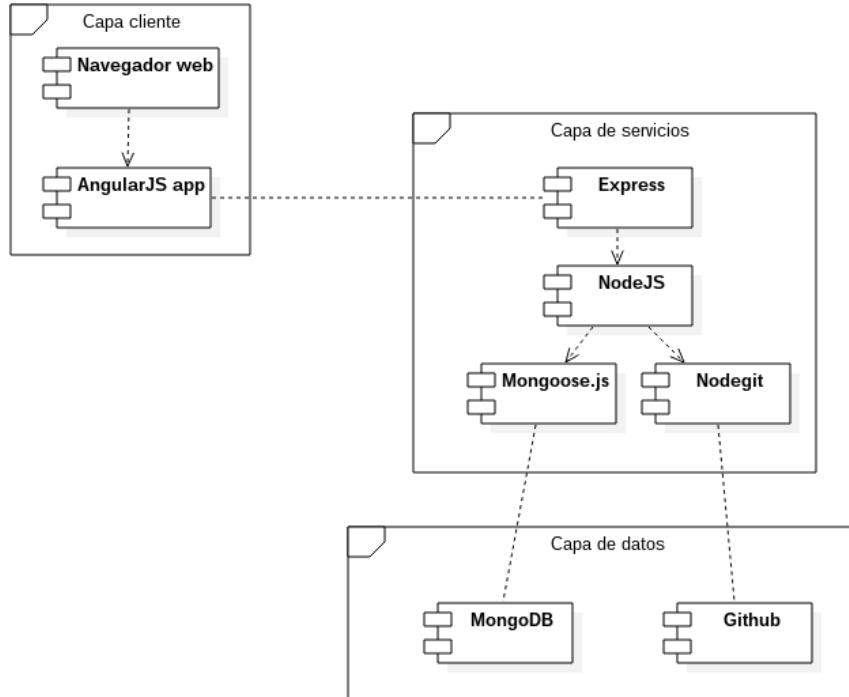


Figura 4. Diagrama de Componentes.

## Vista de Despliegue o Implementación

La vista de implantación se ve reflejada en la figura 5, mostrada a continuación.

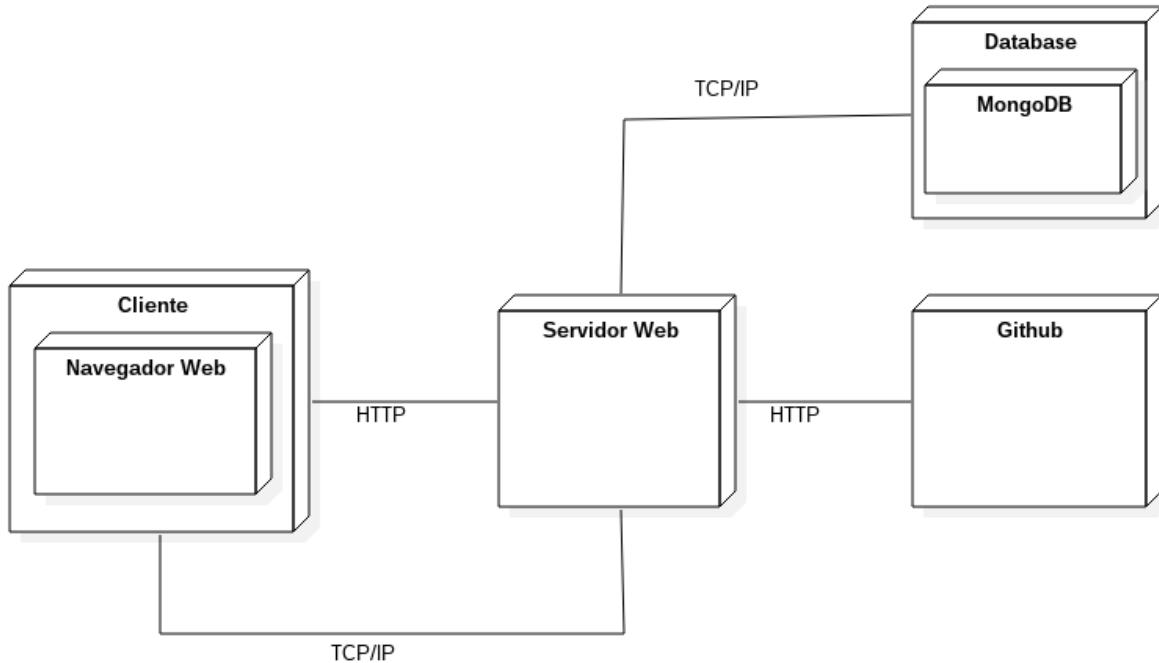


Figura 5. Diagrama de Despliegue.

## Tamaño y Desempeño

- El recurso necesario para la utilización del Asistente para ensamblar aplicaciones es un explorador o navegador de internet.
- El asistente está diseñado para soportar múltiples sesiones de usuario a la vez.

## Calidad

- El desarrollo del Asistente para ensamblar aplicaciones móviles está basado en módulos que otorgan al mismo escalabilidad, permitiendo la incorporación nuevas características fácilmente.
- Una vez que pase a producción, se depende de factores externos como la estabilidad del servidor donde se desenvuelvan los servicios web, por lo que no es posible garantizar con seguridad la disponibilidad de los mismos.
- El ciclo de vida del proyecto de desarrollo del sistema está basado en fases, divididas en iteraciones. Al concluir cada iteración se verifica si se cumplieron los objetivos para

avanzar a la siguiente fase, asegurando así la obtención de un software de buena calidad y bien documentando garantizando su mantenimiento en el tiempo.

## Referencias

Kruchten, P. (Noviembre de 1995). Architectural Blueprints - The "4+1" View Model of Software Architecture. *IEEE Software Magazine*, 42-50. Recuperado el 23 de Septiembre de 2013, de <http://www.cs.ubc.ca/~gregor/teaching/papers/4+1viewarchitecture.pdf>

## **Apéndice D**

### **Especificación de servicios del Asistente para ensamblar aplicaciones móviles**



**SYNERGY-GB**

Especificación de servicios

**Asistente para ensamblar  
aplicaciones móviles**

## Introducción

En este documento se describen los servicios web REST implementados para el backend del Asistente para ensamblar aplicaciones móviles basado en tecnología web. La especificación de estos servicios fue realizada usando la herramienta Swagger.

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/cloneRepo

**POST /cloneRepo**

### Description

Clone a github repository (http clone url)

### Parameters

Name	Located in	Description	Required	Schema
cloneInfo	body		Yes	➡ { destPath: ► string * httpUrl: ► string * }

### Responses

Code	Description
<b>200</b>	Clone successful
<b>400</b>	Bad request

[Try this operation](#)

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/createUser

**POST /createUser**

### Description

Create a new user

### Parameters

Name	Located in	Description	Required	Schema
userInfo	body		Yes	➡ { username: string * password: string * }

### Responses

Code	Description
<b>200</b>	User added
<b>400</b>	Bad request

[Try this operation](#)

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/exec

GET /exec

### Description

Generate the app if doesn't exists, else build the project. Send all the information of the process by websockets.

### Parameters

Name	Located in	Description	Required	Schema
path	query		Yes	↔ string

### Responses

Code	Description
<b>200</b>	Finish operation
<b>400</b>	Bad request

Try this operation

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/genApp

GET /genApp

### Description

Generate apk and send project directory in tar file

### Parameters

Name	Located in	Description	Required	Schema
path	query		Yes	↔ string

### Responses

Code Description

**200** .tar file

**400** Bad request

**404** Path not found

Try this operation

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/generateFolder

**GET /generateFolder**

### Description

Send .tar with project directory

### Parameters

Name	Located in	Description	Required	Schema
path	query		Yes	↔ string

### Responses

Code	Description
<b>200</b>	.tar file
<b>400</b>	Bad request
<b>404</b>	Path not found

[Try this operation](#)

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/getContent

GET /getContent

### Description

Get the content of a file

### Parameters

Name	Located in	Description	Required	Schema
path	query		Yes	↔ string
type	query		No	↔ string

### Responses

Code      Description

**200**      if type == json send json data, else send string data

**400**      Bad request

[Try this operation](#)

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/killApp

GET /killApp

### Description

Kill process

### Parameters

Name	Located in	Description	Required	Schema
pid	query		Yes	↔ string

### Responses

Code	Description
<b>200</b>	Process killed
<b>400</b>	Bad request

[Try this operation](#)

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/loginUser

POST /loginUser

### Description

Login user

### Parameters

Name	Located in	Description	Required	Schema
info	body		Yes	▼ { ⇒     username: ▶ string * password: ▶ string * }

### Responses

Code	Description	Schema
<b>200</b>	Object with user and projects related	▼ { ⇒     username: ▶ string * projects: ▶ [] }
<b>400</b>	Wrong username or password	

Try this operation

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/mongoose\_findApp

GET /mongoose\_findApp

### Description

Find app model

### Parameters

Name	Located in	Description	Required	Schema
app	query		Yes	↔ string

### Responses

Code	Description	Schema
<b>200</b>	Send the app model	▼ { ↔ name: ▶ string * }
<b>400</b>	Bad request	
<b>404</b>	App not found	

Try this operation

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/mongoose\_findGraph

GET /mongoose\_findGraph

### Description

Get application's graph

### Parameters

Name	Located in	Description	Required	Schema
app	query		Yes	↔ string

### Responses

Code	Description	Schema
200	Get application's graph	▼ { ↔ nodes: ▷ [] edges: ▷ [] }
400	Bad request	
404	Project not found	

Try this operation

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/mongoose\_removeElement

**DELETE /mongoose\_removeElement**

### Description

Remove a project

### Parameters

Name	Located in	Description	Required	Schema
app	query		Yes	↔ string
username	query		Yes	↔ string

### Responses

Code                    Description

**200**                    Successful delete

**400**                    Bad request

**404**                    Project not found

[Try this operation](#)

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/mongoose\_setGraph

POST /mongoose\_setGraph

### Description

Save graph of a project

### Parameters

Name	Located in	Description	Required	Schema
app	query		Yes	↔ string
graph	body		Yes	▼ { ↔ graph: ▶ string * }

### Responses

Code                  Description

**200**                  Graph property updated

**400**                  Bad request

Try this operation

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/mongoose\_setProjects

**POST /mongoose\_setProjects**

### Description

Add project to user's project list

### Parameters

Name	Located in	Description	Required	Schema
info	body		Yes	▼ { ↔ username: string project: ► string }

### Responses

Code

Description

**200**

Project name added

**400**

Bad request or project already exists

Try this operation

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/mongoose\_createProject

**POST /mongoose\_createProject**

### Description

Set a new project to a user

### Parameters

Name	Located in	Description	Required	Schema
appModel	body		Yes	▼ { ➡ name: ▶ string }

### Responses

Code	Description
<b>200</b>	Project saved
<b>400</b>	Bad request

[Try this operation](#)

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/mongoose\_updateProject

**PUT /mongoose\_updateProject**

### Description

Update project model

### Parameters

Name	Located in	Description	Required	Schema
info	body		Yes	▼ { username: ► string old_name: ► string model: ► { } }

### Responses

Code	Description
<b>200</b>	Project updated
<b>400</b>	Bad request

[Try this operation](#)

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/moveScreens

**PUT /moveScreens**

### Description

Move screens directory from common directory to app directory

### Parameters

Name	Located in	Description	Required	Schema
info	body		Yes	▼ { base_path: ▶ string copy_path: ▶ string app_path: ▶ string }

### Responses

Code	Description
<b>200</b>	Copy files
<b>400</b>	Bad request

[Try this operation](#)

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/removeComp

**PUT /removeComp**

### Description

Remove component from inventory and remove folder or remove application form prebuilt apps inventory

### Parameters

Name	Located in	Description	Required	Schema
info	body		Yes	↔ { path: ▶ string filename: ▶ string app_name: ▶ string cont: ▶ string }

### Responses

Code	Description
<b>200</b>	App removed or component removed
<b>400</b>	Bad request

[Try this operation](#)

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/runApp

GET /runApp

### Description

Install dependencies and execute application

### Parameters

Name	Located in	Description	Required	Schema
path	query	Application path directory	Yes	↗ string

### Responses

Code Description

**200** Process start. Send PID

[Try this operation](#)

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/setContent

**PUT /setContent**

### Description

Write a file

### Parameters

Name	Located in	Description	Required	Schema
info	body	Content to be written on file and path to file	Yes	<pre>▼ {     path: string     content: string     filename: string }</pre>

### Responses

Code Description

**200** Successful write operation

**400** Bad request

[Try this operation](#)

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/setContentConfig

**PUT /setContentConfig**

### Description

Write a config file

### Parameters

Name	Located in	Description	Required	Schema
info	body	Content to be written on config file and path to file	Yes	<pre>▼ {     path: string     content: string     filename: string }</pre>

### Responses

Code Description

**200** Successful write operation

**400** Bad request

[Try this operation](#)

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/setInventario

**PUT /setInventario**

### Description

Modify inventory file

### Parameters

Name	Located in	Description	Required	Schema
info	body	Content to be written on inventory file and path to file	Yes	<pre>▼ {     path: string     content: string     filename: string }</pre>

### Responses

Code Description

**200** Successful write operation

**400** Bad request

Try this operation

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/setMetadata

**PUT /setMetadata**

### Description

Modify metadata file

### Parameters

Name	Located in	Description	Required	Schema
info	body	Content to be written on metadata file and path to file	Yes	▼ { path: string content: string filename: string screens_path: string }

### Responses

Code                  Description

**200**                  Successful write operation

**400**                  Bad request

[Try this operation](#)

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/uploadFiles

**POST /uploadFiles**

### Description

Upload resources

### Parameters

Name	Located in	Description	Required	Schema
path	query	Upload resources for customize apps	Yes	➡ string

### Responses

Code Description

**200** Successful upload

**400** Bad request

[Try this operation](#)

# Kiraso Services

Kiraso services documentation

**Version** 1.0.0

## Paths

/dirTree

GET /dirTree

### Description

Get working directory

### Parameters

Name	Located in	Description	Required	Schema
path	query		Yes	↔ string

### Responses

Code	Description	Schema
<b>200</b>	Working directory structure	↔ [ ▶ Directory { } ]
<b>400</b>	Bad request	

Try this operation

## Models

### Directory

```
▼Directory {  
    path:    ▶ string *  
    text:    ▶ string *  
    type:    ▶ string *  
    submenu: ▶ []  
}
```

## **Apéndice E**

### **Matriz de pruebas**



**SYNERGY-GB**

Matriz de pruebas

**Asistente para ensamblar  
aplicaciones móviles**

## Introducción

Este documento provee la matriz de pruebas funcionales realizadas sobre el Asistente para ensamblar aplicaciones móviles basado en tecnología web durante el desarrollo del producto para comprobar su buen funcionamiento.

Para efectos de este documento únicamente se describirán las pruebas realizadas. Estas deben refinarse cuando la empresa realice la fase de transición, de modo que se pueda certificar el cumplimiento de todos los requerimientos funcionales de la aplicación.

## Synergy-GB +Calidad

*Pruebas integrales sobre la aplicación Asistente para ensamblar aplicaciones móviles basado en tecnología web*

ID	Caso de Prueba	Resultado	Condiciones y Pasos	Datos de entrada	Resultado esperado
<b>1</b>	<b>Gestión de usuarios</b>				
P1-001	Iniciar sesión ingresando usuario y contraseña válidos	Pendiente	1.- Ingresar usuario y contraseña válidos 2.- Presionar Botón: Login. 3.- La aplicación mostrará los proyectos del usuario.	Usuario: Contraseña:	Inicio de sesión exitoso, redirecciona al listado de aplicaciones.
P1-002	Iniciar sesión ingresando usuario inválido	Pendiente	1.- Ingresar usuario inválido 2.- Ingresar contraseña válida 3.- Presionar Botón: Login.	Usuario: Contraseña:	Usuario o clave inválido
P1-003	Iniciar sesión ingresando contraseña inválida	Pendiente	1.- Ingresar usuario válido 2.- Ingresar contraseña Inválida 3.- Presionar Botón: Login.	Usuario: Contraseña:	Usuario o clave inválido

<b>P1-004</b>	Iniciar sesión omitiendo usuario	<b>Pendiente</b>	1.- Omitir Usuario 2.- Ingresar contraseña válida 3.- Presionar Botón: Login. 4.- Mostrara un mensaje de error	Usuario: Contraseña:	Campo requerido
<b>P1-005</b>	Iniciar sesión omitiendo contraseña	<b>Pendiente</b>	1.- Ingresar usuario válido 2.- Omitir Contraseña 3.- Presionar Botón: Login. 4.- Mostrara un mensaje de error	Usuario: Contraseña:	Campo requerido
<b>P1-006</b>	Iniciar sesión omitiendo usuario y contraseña	<b>Pendiente</b>	1.- Omitir Usuario y Contraseña 2.- Presionar Botón: Login. 3.- Mostrara un mensaje de error	Usuario: Contraseña:	Campo requerido
<b>P1-007</b>	Iniciar sesión ingresando una longitud de usuario de rango menor al aceptado	<b>Pendiente</b>	1.- Ingresar usuario con rango menor a: 4 Dígitos/Letras 2.- Ingresar contraseña válida 3.- Presionar Botón: Login. 4.- Mostrara un mensaje de error		Nombre de usuario muy corto

<b>P1-008</b>	Iniciar sesión ingresando una longitud de contraseña de rango menor al aceptado	<b>Pendiente</b>	1.- Ingresar usuario válido 2.- Ingresar contraseña con rango menor a: 6 Dígitos/Letras 3.- Presionar Botón: Login. 4.- Mostrará un mensaje de error		Contraseña muy corta
<b>P1-009</b>	Iniciar sesión sin disponibilidad de los servicios	<b>Pendiente</b>	1.- Solicitar detener el servidor. 2.- Ingresar usuario y contraseña válidos. 3.- Presionar Botón: Aceptar	Usuario: Contraseña:	
<b>P1-010</b>	Cerrar Sesión	<b>Pendiente</b>	1.- Autenticarse satisfactoriamente 2.- Seleccionar Botón de cierre de sesión 3.- Confirmar la acción		Cierre de sesión exitoso
<b>P1-011</b>	Registrar usuario ingresando usuario y contraseñas válidas	<b>Pendiente</b>	1.- Ingresar usuario y contraseñas válidas 2.- Presionar Botón: Sign Up. 3.- La aplicación mostrará los proyectos del usuario.	Usuario: Contraseña: Confirmar contraseña:	Registro de usuario exitoso.

P1-012	Registrar usuario ingresando contraseñas distintas	Pendiente	1.- Ingresar usuario válido 2.- Ingresar contraseña válida 3.- Ingresar confirmación de contraseña inválida 4.- Presionar Botón: Sign up	Usuario: Contraseña: Confirmar contraseña:	Contrasenas no coinciden
P1-013	Registrar usuario ingresando usuario válido pero existente y contraseñas válidas	Pendiente	1.- Ingresar usuario y contraseñas válidas 2.- Presionar Botón: Sign Up. 3.- La aplicación mostrará los proyectos del usuario.	Usuario: Contraseña: Confirmar contraseña:	Usuario ya existe
P1-014	Registrar usuario omitiendo algún campo (o todos)	Pendiente	1.- Omitir algún campo (o todos) 2.- Presionar Botón: Sign up. 3.- Mostrará un mensaje de error	Usuario: Contraseña: Confirmar contraseña:	Campo requerido
P1-015	Registrar usuario ingresando una longitud de usuario de rango menor al aceptado	Pendiente	1.- Ingresar usuario con rango menor a: 4 Dígitos/Letras 2.- Ingresar contraseña válida 3.- Presionar Botón: Sign up. 4.- Mostrará un mensaje de error		Nombre de usuario muy corto

P1-016	Registrar usuario ingresando una longitud de contraseña o de confirmación de contraseña de rango menor al aceptado	Pendiente	1.- Ingresar usuario válido 2.- Ingresar contraseña con rango menor a: 6 Dígitos/Letras 3.- Presionar Botón: Login. 4.- Mostrara un mensaje de error		Contraseña muy corta
<b>2 Gestión de aplicaciones</b>					
P2-001	Crear aplicación	Pendiente	1.- Ingresar nombre de la aplicación nuevo	Nombre de la aplicación:	Se crea la aplicación y se redirecciona a la vista gráfica de asistente
P2-002	Crear aplicación con nombre existente	Pendiente	1.- Ingresar nombre de la aplicación existente	Nombre de la aplicación:	Indica que la aplicación ya existe.
P2-003	Crear aplicación dejando el nombre vacío	Pendiente	1.- Dejar campo de nombre de aplicación en blanco	Nombre de la aplicación:	Campo requerido
P2-004	Crear aplicación sin disponibilidad de servicios	Pendiente	1.- Deshabilitar servicios 2.- Ingresar nombre de aplicación. 3.- Presionar botón guardar		Muestra mensaje: aplicación guardada Funcionalidad: Hacer clic fuera del popup
P2-005	Guardar aplicación	Pendiente	1.- Presionar botón de guardar		Muestra mensaje: aplicación guardada Funcionalidad: Hacer clic fuera del popup

<b>P2-006</b>	Guardar aplicación sin servicios disponibles	<b>Pendiente</b>	1.- Deshabilitar servicios 2.- Presionar botón de guardar		Muestra mensaje: error guardando aplicación Funcionalidad: Hacer clic fuera del popup
<b>P2-005</b>	Eliminar aplicación	<b>Pendiente</b>	1.- Presionar botón de eliminar aplicación		Funcionalidad: Sigue confirmación, si se confirma borra la aplicación
<b>P2-006</b>	Eliminar aplicación sin servicios disponibles	<b>Pendiente</b>	1.- Deshabilitar servicios 2.- Presionar botón de eliminar aplicación		Funcionalidad: Sigue confirmación, si se confirma no hace nada
<b>P2-007</b>	Clonar aplicación	<b>Pendiente</b>	1.- Se selecciona una aplicación en la sección de aplicaciones pre construidas	Aplicación seleccionada:	Funcionalidad: Sigue confirmación, si se confirma carga el grafo de la aplicación
<b>P2-008</b>	Clonar aplicación sin servicios disponibles	<b>Pendiente</b>	1.- Deshabilitar servicios 2.- Se selecciona una aplicación en la sección de aplicaciones pre construidas	Aplicación seleccionada:	Funcionalidad: Sigue confirmación, si se confirma no hace nada
<b>P2-009</b>	Agregar componente	<b>Pendiente</b>	1.- Se selecciona un componente de la lista de componentes	Nombre de componente:	Funcionalidad: Crea un nodo con su respectivo nombre en el grafo

<b>P2-010</b>	Agregar componente sin servicios disponibles	<b>Pendiente</b>	1.- Se selecciona un componente de la lista de componentes	Nombre de componente:	Funcionalidad: Crea un nodo con su respectivo nombre en el grafo, si se selecciona da error de metadata
<b>P2-011</b>	Eliminar componente	<b>Pendiente</b>	1.- Presionar botón de eliminar componente		Funcionalidad: Solicita confirmación, si se confirma borra el componente
<b>P2-012</b>	Eliminar componente sin servicios disponibles	<b>Pendiente</b>	1.- Deshabilitar servicios 2.- Presionar botón de eliminar componente		Funcionalidad: Solicita confirmación, si se confirma no hace nada
<b>P2-013</b>	Configurar componente: datos de pantalla	<b>Pendiente</b>	1.- Seleccionar un nodo del grafo 2.- Seleccionar la sección de "screen" 3.- Llenar el formulario	Componente seleccionado: Screen name: Default screen:	Muestra mensaje: información de screen agregada Funcionalidad: Hacer clic fuera del popup
<b>P2-014</b>	Configurar componente: datos de pantalla sin disponibilidad de servicios	<b>Pendiente</b>	1.- Deshabilitar servicios 1.- Seleccionar un nodo del grafo 2.- Seleccionar la sección de "screen" 3.- Llenar el formulario	Componente seleccionado: Screen name: Default screen:	Muestra alerta al seleccionar el nodo: Error cargando metadata Muestra mensaje: información de screen agregada Funcionalidad: Hacer clic fuera del popup

P2-015	Configurar componente: datos de parámetros de la pantalla(si los requiere)	Pendiente	1.- Seleccionar un nodo del grafo 2.- Seleccionar la sección de "params" 3.- Llenar el formulario	Componente seleccionado: Detail <b>*Parámetros dependen del componente*</b>	Muestra mensaje: información de params agregada Funcionalidad: Hacer clic fuera del popup
P2-016	Configurar componente: datos de parámetros de la pantalla (si los requiere) sin disponibilidad de servicios	Pendiente	1.- Deshabilitar servicios 2.- Seleccionar un nodo del grafo 3.- Seleccionar la sección de "params" 4.- Llenar el formulario	Componente seleccionado: Detail <b>*Parámetros dependen del componente*</b>	Muestra alerta al seleccionar el nodo: Error cargando metadata Muestra mensaje: información de params agregada Funcionalidad: Hacer clic fuera del popup
P2-017	Configurar componente: datos de la fuente de datos de la pantalla(si los requiere)	Pendiente	1.- Seleccionar un nodo del grafo 2.- Seleccionar la sección de "data" 3.- Llenar el formulario	Componente seleccionado: Detail <b>*Fuente de datos dependen del componente*</b>	Muestra mensaje: información de data agregada Funcionalidad: Hacer clic fuera del popup
P2-018	Configurar componente: datos de la fuente de datos de la pantalla (si los requiere) sin disponibilidad de servicios	Pendiente	1.- Deshabilitar servicios 2.- Seleccionar un nodo del grafo 3.- Seleccionar la sección de "data" 4.- Llenar el formulario	Componente seleccionado: Detail <b>*Fuente de datos dependen del componente*</b>	Muestra alerta al seleccionar el nodo: Error cargando metadata Muestra mensaje: información de data agregada Funcionalidad: Hacer clic fuera del popup

<b>P2-019</b>	Consultar componente	<b>Pendiente</b>	1.- Seleccionar opción de consulta de componente de aplicación		Re-direcciona a repositorio Github con la información del componente
<b>P2-020</b>	Clonar componente	<b>Pendiente</b>	1.- Seleccionar opción agregar componente 2.- Llenar formulario	Nombre del componente: Tipo del componente: Url(https):	Muestra mensaje: Clonando repositorio Funcionalidad: Una vez que cierra el popup ya se ha agregado el componente y se cierra el formulario.
<b>P2-021</b>	Clonar componente sin servicios disponibles	<b>Pendiente</b>	1.- Seleccionar opción agregar componente 2.- Llenar formulario	Nombre del componente: Tipo del componente: Url(https):	Muestra mensaje: Error clonando e repositorio Funcionalidad: Hacer clic fuera del popup
<b>P2-022</b>	Ejecutar aplicación	<b>Pendiente</b>	1.- Ir a sección de vista previa 2.- Esperar que cargue directorio de aplicación 3.- Seleccionar opción de ejecutar aplicación		Muestra mensaje por consola de los procesos en ejecución

P2-023	Ejecutar aplicación sin servicios disponibles	Pendiente	1.- Ir a sección de vista previa 2.- Esperar que cargue directorio de aplicación 3.- Deshabilitar servicios 4.- Seleccionar opción de ejecutar aplicación		No muestra mensajes por consola
P2-024	Mostrar vista previa	Pendiente	1.- Ir a sección de vista previa 2.- Esperar que cargue directorio de aplicación 3.- Seleccionar opción de ejecutar aplicación 4.- Esperar que la ejecución culmine satisfactoriamente 5.- Seleccionar opción de recarga de vista previa		Muestra vista previa de la aplicación

P2-025	Mostrar vista previa sin servicios disponibles	Pendiente	1.- Ir a sección de vista previa 2.- Esperar que cargue directorio de aplicación 3.- Seleccionar opción de ejecutar aplicación 4.- Esperar que la ejecución culmine satisfactoriamente 5.- Deshabilitar servicios 5.- Seleccionar opción de recarga de vista previa		No muestra la vista previa de la aplicación
P2-025	Mostrar vista previa sin ejecución	Pendiente	1.- Ir a sección de vista previa 2.- Esperar que cargue directorio de aplicación 3.- Seleccionar opción de recarga de vista previa		No muestra la vista previa de la aplicación
P2-026	Listar Archivos de aplicación	Pendiente	1.- Ir a sección de vista previa 3.- Esperar que cargue directorio de aplicación		Muestra mensaje por consola de los procesos en ejecución

<b>P2-027</b>	Listar Archivos de aplicación sin servicios disponibles	<b>Pendiente</b>	1.- Deshabilitar servicios 2.- Ir a sección de vista previa 3.- Esperar que cargue directorio de aplicación		No muestra mensajes de consola ni carga lista de archivos
<b>P2-028</b>	Ver archivos	<b>Pendiente</b>	1.- Ir a sección de vista previa 2.- Esperar que cargue directorio de aplicación 3.- Seleccionar archivo		Carga el contenido del archivo en el editor web
<b>P2-029</b>	Ver archivos sin servicios disponibles	<b>Pendiente</b>	1.- Ir a sección de vista previa 2.- Esperar que cargue directorio de aplicación 3.- Deshabilitar servicios 3.- Seleccionar archivo		Muestra mensaje: Error al cargar el contenido de archivo
<b>P2-030</b>	Agregar recursos	<b>Pendiente</b>	1.- Ir a sección de vista previa 2.- Esperar que cargue directorio de aplicación 3.- Seleccionar opción de cargar recursos 4.- Seleccionar archivo a cargar	Archivo con formato .tar o .zip	Muestra ventana de selección de archivo

P2-031	Agregar recursos con formato erróneo	Pendiente	1.- Ir a sección de vista previa 2.- Esperar que cargue directorio de aplicación 3.- Seleccionar opción de cargar recursos 4.- Seleccionar archivo a cargar	Archivo con formato diferente a .zip o .tar	No permite agregar el recurso
P2-032	Agregar recursos sin servicios disponibles	Pendiente	1.- Ir a sección de vista previa 2.- Esperar que cargue directorio de aplicación 3.- Seleccionar opción de cargar recursos 4.- Deshabilitar recursos 5.- Seleccionar archivo a cargar	Archivo con formato .tar o .zip	No permite agregar el recurso
P2-033	Descargar estructura de archivos	Pendiente	1.- Seleccionar opción descargar 2.- Seleccionar nombre y carpeta de descarga		Muestra mensaje mientras descarga archivos

<b>P2-034</b>	Descargar estructura de archivos sin servicios disponibles	<b>Pendiente</b>	1.- Deshabilitar servicios 2.- Seleccionar opción descargar 3.- Seleccionar nombre y carpeta de descarga		Muestra mensaje: error descargando archivo
<b>P2-035</b>	Descargar estructura de archivos sin haber creado la estructura de archivos	<b>Pendiente</b>	1.- Crear aplicación 2.- Seleccionar opción descargar		Muestra mensaje: error descargando archivo

# **Apéndice F**

## **Manual Kiraso**



**SYNERGY-GB**

**Manual Kiraso**

**Asistente para ensamblar  
aplicaciones móviles**

## Introducción

Este documento provee un pequeño manual con las instrucciones para instalar el ambiente del Asistente para ensamblar aplicaciones móviles basado en tecnología web, así como también, una guía para generar los archivos de metadatos necesarios de los componentes.

# Manual Kiraso

## Guía implantación

1. Servidor con ambiente de Kiraso instalado.
2. Instalar MongoDB.
3. Dependencias necesarias a instalar por NPM:
  - a. mongoose
  - b. nodegit
  - c. mkdirp
  - d. lodash
  - e. multer
  - f. socket.io
4. Ubicar archivos de metadata y de inventarios de componentes en el servidor.
5. Una vez instalado el servidor cambiar en el Wizard el archivo custom.controller.js el url base de los servicios en la variable \$rootScope.url
6. En código del Wizard cambiar los path de los requests respectivos a la ubicación de los archivos en el servidor.

Hacer bower install en el código del asistente de Kiraso.



- Para ejecutar el código cliente hacer gulp serve
- Para ejecutar el código del servidor por ahora ejecutar DEBUG=sgb-kiraso-services:\* npm start

## Guía para generar metadata de componentes

La metadata de los componentes sirve para que el asistente de Kiraso pueda definir los parámetros del componente. Para generar la metadata de los componentes se deben seguir las reglas de definición de esquemas de angular schema form. Se puede consultar en <http://schemaform.io/examples/bootstrap-example.html>.

La estructura de los archivos es la siguiente:

1. Nombre del componente
2. Tipo de componente
3. Una lista de parámetros donde cada parámetro tiene:
  - a. Nombre del parámetro
  - b. Tipo de dato
  - c. Descripción del parámetro
  - d. Título de la etiqueta
  - e. En caso de ser necesario se agrega atributo extra:
    - i. Para arreglos el atributo elements con la lista de elementos de entrada que tendrá el arreglo. Ver ejemplo en demo de angular schema form
    - ii. Para opciones de un select usar el atributo options con la lista de opciones del dropdown. Ver ejemplo en demo de angular schema form

### Metadata de componentes de Synergy-GB

sgb-screen-list	Metadata - lista
sgb-screen-menu	Metadata - menu
sgb-screen-detail	Metadata - details
sgb-screen-dashboard	Metadata - dashboard
sgb-screen-group-list	Metadata - group list
sgb-screen-login	Metadata - login
sgb-screen-tour	Metadata - tour



Los archivos de metadata siempre deben nombrarse "metadata.json"

La ubicación de los archivos de metadata debe ser la siguiente:

- Si son componentes de la empresa, y pertenecen al inventario de la empresa en el servidor, debe estar en un directorio donde se encuentre este archivo y ser especifica la ruta en el inventario de componentes de la empresa.
- Si son componentes propios debe estar en la raíz de la estructura de archivos del componente.