



**IES AUGUSTO GONZALEZ DE
LINARES**

DEPARTAMENTO DE INFORMATICA

PROYECTO CHIKARA

0492 - PROYECTO DE DESARROLLO DE APLICACIONES MULTIPLATAFORMA

CICLO FORMATIVO DE GRADO SUPERIOR


DESARROLLO DE APLICACIONES MULTIPLATAFORMA

2023/2024

Díez de Paulino, Albano

Espinosa García, Daniel

Gutiérrez Valverde, Ramiro

	Ciclo: Desarrollo de Aplicaciones Multiplataforma
	Modulo: Proyecto De Desarrollo De Aplicaciones Multiplataforma
	Título: Proyecto Chikara

Histórico de modificaciones

Versión	Fecha	Hecho	Descripción
1	02/04/2024	Documento Inicial	Documentación de proyecto
2	05/04/2024	Segunda Entrega	Modelo de datos
3	15/05/2024	Tercera Entrega	Diseño de la interfaz

Licencia del documento



El presente documento se encuentra sujeto a la licencia Creative Commons BY 4.0 Internacional, si no dispone de una copia visite el siguiente sitio web

<https://creativecommons.org/licenses/by/4.0/>



ÍNDICE

1.	RESUMEN.....	6
2.	PALABRAS CLAVE.....	8
3.	INTRODUCCIÓN	10
4.	OBJETIVOS.....	12
4.1.	Requisitos funcionales	12
4.2.	Requisitos no funcionales	13
5.	ANÁLISIS DEL CONTEXTO	16
5.1.	Análisis del contexto.....	16
5.2.	Innovación	17
6.	PLANIFICACIÓN	19
6.1.	Diagrama de Gantt.....	20
6.2.	Recursos técnicos y humanos	24
7.	DISEÑO	26
7.1.	Mockup	26
7.2.	Casos de Uso.....	28
7.3.	Modelos de datos	29
7.3.1.	PostgreSQL.....	29
7.3.2.	MongoDB	30
7.4.	Estructura del proyecto	30
7.4.1.	Arquitectura Back-end.....	30
7.4.2.	Arquitectura Front-end	35
7.5.	Despliegue	36
8.	IMPLEMENTACIÓN	40
9.	PUESTA EN MARCHA	42
10.	PLAN DE PRUEBAS	44
11.	PLAN DE EMPRESA.....	46
11.1.	IDENTIFICACIÓN	46



11.1.1.	Historia de la empresa.....	46
11.1.2.	Cultura empresarial.....	46
11.1.3.	Identificación del equipo	50
11.1.4.	Descripción actividad empresarial.....	50
11.2.	ANALISIS DE MERCADO.....	50
11.2.1.	Análisis Macroentorno.....	50
11.2.2.	Datos generales del sector.....	50
11.2.3.	Clientes Potenciales	50
11.2.4.	Plan de Marketing.....	50
11.2.5.	DAFO	50
11.3.	PLAN TECNICO-PRODUCTIVO	50
11.3.1.	Servicios Ofrecidos.....	50
11.3.2.	Tabla de Precios.....	50
11.3.3.	Distribución	50
11.4.	PLAN DE RECURSOS HUMANOS.....	50
11.4.1.	Perfiles profesionales.....	50
11.4.2.	Organigrama	50
11.4.3.	Política Salarial.....	50
11.4.4.	Política de Teletrabajo	50
11.4.5.	Política acoso Laboral.....	50
11.5.	PLAN DE INVERSIONES	50
11.5.1.	Localización Sede.....	50
11.5.2.	Inmovilizado intangible.....	50
11.5.3.	Inmovilizado Material.....	50
11.5.4.	Presupuesto Detallado de Gastos.....	50
11.5.5.	Resumen de Inversiones	50



11.5.6.	Fuentes de financiación	50
11.6.	OBLIGACIONES EMPRESARIALES	50
11.6.1.	Forma jurídica.....	50
11.6.2.	Ventajas	50
11.6.3.	Desventajas	51
11.6.4.	Pasos que rellenar para formar la empresa	51
11.6.5.	Obligaciones Laborales.....	51
11.6.6.	Obligaciones sobre prevención Laboral	51
12.	CONCLUSIONES	53
13.	BIBLIOGRAFÍA.....	55
14.	ÍNDICES	57
14.1.	Índice de Tablas	57
14.2.	Índice de Ilustraciones	57
ANEXOS	59
Anexo I	Registro Horario.....	59
Anexo II	Estatutos de Empresa	59
Anexo III	Modelo 036.....	59
PREGUNTAS FRECUENTES	61
1.	¿Por qué se ha usado un modelo de datos relacional y un modelo de datos no-relacional conjuntamente?.....	61
2.	¿Cómo se relacional ambos modelos de datos usados?	61
3.	¿Por qué hemos usado el servicio cloud Azure de Microsoft sobre otras alternativas?.....	61
4.	¿Por qué usamos una arquitectura de microservicios en vez de una monolítica?	61
5.	¿El desarrollo móvil es solo para dispositivos con SO Android, tenemos planes para un desarrollo para SO iOS? ¿Qué lenguaje usaríamos?.....	61
6.	¿Si se deja planteado un desarrollo para múltiples dispositivos porque no hemos usado un framework multiplataforma?	61

RESUMEN



1. RESUMEN

Chikara es una aplicación multiplataforma que, en la fase actual del proyecto se concreta en una aplicación móvil para dispositivos Android, pero que puede ser implementada para otros sistemas operativos y plataformas.

Se trata de una red social que busca convertirse en un espacio de motivación personal en la que los usuarios pueden obtener inspiración de otros usuarios, o crear su propio contenido inspirador. En Chikara ponemos la fuerza que necesitas en un momento de flaqueza dentro de tu bolsillo. Además, podrás regalar fuerza a tus contactos (nakamas), creando contenido pensado para ellos.

Los chiks (nombre que le hemos puesto a dicho contenido), se componen de imágenes y textos con gran carga emotiva para el usuario, que en conjunto ofrecen un impulso en una determinada dirección.

A diferencia de otras redes sociales, que se nutren de tu atención, drenando tu energía y consumiéndote entre caminos sinuosos, Chikara te proporciona foco, dirección e impulso. Puedes pasar el tiempo que quieras en chikara, pero ese tiempo se traducirá en motivación, fuerza, Chikara.

PALABRAS

CLAVE



2. PALABRAS CLAVE

A continuación, se detalla una relación de los términos que se irán usando a lo largo de este documento:

- **Requisito funcional:** Requisito que define el comportamiento interno del desarrollo, como cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas.
- **Requisito no funcional:** Se trata de requisitos que no se refieren directamente a las funciones específicas suministradas por el sistema, sino a las propiedades del sistema, como la estética o la seguridad.
- **Aplicación Multiplataforma:** Aplicación que puede ser ejecutada desde diferentes plataformas y/o Sistemas Operativos.
- **Base de datos:** identifica dónde se encuentra alojada la información del proyecto.
- **Front-end:** interfaz visual del sistema que se desarrolle.
- **Back-end:** operaciones y lógicas que realiza en sistema desarrollado entre la base de datos y el front-end.
- **API Rest:** Interfaz de programación de aplicaciones que permite la comunicación entre ordenadores.
- **Cloud o Cloud Computing:** Uso de una red de servidores remotos bajo demanda conectados a internet para procesar, almacenar o interconectar.
- **Microservicios:** Arquitectura del software que representa la separación de los servicios de forma independiente.
- **Red Social:** Estructura social compuesta por un conjunto de actores con una o más relaciones definidas entre ellos.
- **Nakama:** Palabra japonesa para referirse a amigo, que será utilizada en la aplicación para nombrar a tus contactos.
- **Chikara:** Palabra japonesa que significa fuerza, utilizada para referirse al nombre de la aplicación
- **Chik:** Adaptación del término chikara que será utilizado para referirse a cada uno de los bloques de contenido que los usuarios crean a modo de motivación. Estará compuesto de uno o varios elementos. Pueden ser de texto o imágenes.
- **Chiks:** Plural de chik. Se refiere al conjunto de estos.

INTRODUCCIÓN



3. INTRODUCCIÓN

El ser humano es increíble, a todos los niveles. Nunca deja de sorprendernos con sus capacidades. Biológicamente no somos muy diferentes de la mayoría de los mamíferos que existen, pero a diferencia de ellos poseemos una mente superior.

En la mayoría de las ocasiones ésta trabaja a nuestro favor. Una herramienta capaz de encontrar soluciones a los problemas más diversos imaginables y, a la vez, de conseguir lo inimaginable en la búsqueda de la excelencia y la perfección. Pero, en ocasiones la mente deja de ser nuestra herramienta, nuestro aliado primordial, y se convierte en un obstáculo a superar. La falta de motivación, la pérdida de foco, impulso, dirección, como quiera llamarse se traduce en fallo, abandono, desidia, sensación de impotencia.

Nosotros en Café Con Palito vemos esa llama como la fuerza necesaria para llevar a buen término los proyectos vitales y profesionales. Esta aplicación es un brindis, un reconocimiento a esa fuerza y a su importancia en la historia de la humanidad.

Chikara y los chik.

Si hay una cultura que pueda destacar sobre el resto en su dedicación, tesón y respeto por la persistencia y el trabajo duro esa es la japonesa. Por eso el nombre de nuestra aplicación va a ser **CHIKARA**, la palabra japonesa para la fuerza. Su kanji, además, será el icono.

Un **CHIK** es un conjunto de imágenes y textos que contienen la fuerza, la energía que una persona necesita para superar cualquier bache y centrarse de nuevo en su objetivo. No es solamente motivación. Es tu motivación.

Crea un CHIK para cuando olvides por qué estás estudiando tan duro, entrenando, dejando de fumar, invirtiendo tanto tiempo en ese proyecto que no parece avanzar... Los esfuerzos parecen menores cuando se tiene la fuerza necesaria.

OBJETIVOS



4. OBJETIVOS

Los objetivos del presente punto son:

1. Definir exhaustivamente los requisitos deseables para el sistema que se pretende desarrollar.
2. Determinar el alcance definitivo del proyecto para adecuarlo a las horas de desarrollo contratadas.

Se definirán, por tanto, los requisitos que debe cumplir el sistema en su conjunto más amplio, sobrepasando los límites de tiempo de desarrollo del proyecto, especificando los detalles técnicos necesarios para llevarlos a cabo (fuentes u orígenes de datos, procesos de transformación, algoritmia, etc.) y, posteriormente, se etiquetarán conforme a su consideración dentro del proyecto como **[dentro de alcance]** o **[fuera del alcance]**.

Los requisitos **[dentro de alcance]** serán abordados dentro de los límites del proyecto actual y los **[fuera de alcance]** no se llevarán a cabo en el proyecto actual, pero se tendrán en cuenta en el diseño con el fin de permitir la escalabilidad del sistema.

En el supuesto de no contar con la suficiente información con relación a algún punto, se marcará como **[pendiente de análisis]**. Es posible que a lo largo del desarrollo del proyecto algunos o todos los elementos marcados con esta etiqueta se analicen en profundidad, lo que puede provocar su inclusión como **[dentro de alcance]**.

Si no llegaran a analizarse, o una vez analizados no suponen un cambio en las prioridades del proyecto, estos puntos quedaran **[fuera de alcance]** y se contemplarían en futuras ampliaciones del proyecto.

4.1.Requisitos funcionales

En este apartado se van a determinar todos aquellos requisitos del sistema que son determinantes para la funcionalidad, que influyen en el mantenimiento, usabilidad o escalabilidad.



- Aplicación Android
 - Sistema de login [dentro de alcance].
 - Sistema de registro [dentro de alcance].
 - Visualización de tus chiks [dentro de alcance].
 - Buscador de chiks [dentro de alcance].
 - Añadir nuevo chik [dentro de alcance].
 - Eliminar y modificar chiks [dentro de alcance].
- Api Rest
 - Comunicación PostgreSQL [dentro de alcance].
 - Comunicación MongoDB [dentro de alcance].
 - Securización (JWT) [dentro de alcance].
 - Comunicación Blob Storage [dentro de alcance].
- Bases de datos
 - Sistema gestor de bases de datos PostgreSQL [dentro de alcance].
 - Sistema gestor de bases de datos MongoDB [dentro de alcance].
- Aplicación de escritorio
 - Sistema de login [fuera de alcance].
 - Sistema de registro [fuera de alcance].
 - Visualización de tus chiks [fuera de alcance].
 - Buscador de chiks [fuera de alcance].
 - Añadir nuevo chik [fuera de alcance].
 - Eliminar y modificar chiks [fuera de alcance].

4.2.Requisitos no funcionales

En este apartado se van a determinar todos aquellos requisitos del sistema que no son determinantes para la funcionalidad, pero que influyen en el mantenimiento, usabilidad, escalabilidad, como puedan ser arquitecturas, necesidades de recarga de la información, definición de perfiles, etc.

- Aplicación Android
 - Sistema de nakamas [dentro de alcance].



- Sistema de recuperación de contraseña **[dentro de alcance]**.
 - Sistema de invitación **[fuera de alcance]**.
 - Sistema de likes **[fuera de alcance]**.
 - Etiquetado **[fuera de alcance]**.
 - Comentarios **[fuera de alcance]**.
 - Foto de usuario **[pendiente de análisis]**.
- Api Rest
 - Sistema de notificación de nuevas conexiones por correo **[dentro de alcance]**.
 - Lógica de recuperación de contraseña **[dentro de alcance]**.
 - Lógica de nakamas **[dentro de alcance]**.
 - Lógica de invitaciones **[fuera de alcance]**.
 - Lógica de likes **[fuera de alcance]**.
 - Lógica de etiquetado **[fuera de alcance]**.
 - Lógica de comentarios **[fuera de alcance]**.
 - Lógica foto de usuario **[pendiente de análisis]**.
- Aplicación de escritorio
 - Sistema de nakamas **[fuera de alcance]**.
 - Sistema de recuperación de contraseña **[fuera de alcance]**.
 - Sistema de invitación **[fuera de alcance]**.
 - Sistema de likes **[fuera de alcance]**.
 - Etiquetado **[fuera de alcance]**.
 - Comentarios **[fuera de alcance]**.
 - Foto de usuario **[pendiente de análisis]**.

ANÁLISIS DEL CONTEXTO



5. ANÁLISIS DEL CONTEXTO

5.1. Análisis del contexto

En la actualidad, haciendo un repaso de las principales redes sociales del mercado, no encontramos ninguna que realice la actividad que proponemos desde Café con Palito. Si bien es cierto que algunas aplicaciones existentes, como Facebook o Instagram de Meta o TikTok de ByteDance, ofrecen funcionalidades que pueden albergar contenido similar al que nosotros desarrollamos, nunca es exclusivo, y desaparece, no consigue destacar entre el resto de contenido disminuyendo o anulando su eficacia.

Facebook, Instagram y TikTok, en un abanico de edades que abarcan todas las franjas posibles, atraen a un tipo de usuario que llega a consumir contenido variado, la mayoría pasivo, y a crear contenido para que otros consuman. Se basa en un sistema de recompensas y de algoritmos que ofrecen al usuario contenido dirigido, utilizando datos de este, que son recogidos por la propia aplicación o comprados por la empresa que las despliega. Aquí no hay más interés que el de mantener al usuario conectado y consumiendo anuncios, ya que se trata de aplicaciones gratuitas, con un alto mantenimiento y consumo de recursos, que han de responder ante su accionariado.

Un análisis DAFO del proyecto podría resumirse de la siguiente manera:

Debilidades:

- El mercado está sobresaturado de herramientas de autoayuda, redes sociales y aplicaciones en general, que disponen de un presupuesto mucho mayor que el nuestro, por lo que puede resultar muy complicado hacerse un hueco.
- La aplicación adquiere atractivo a medida que más y más usuarios se la descargan y la utilizan, por lo que en un principio puede no resultar llamativa.

Amenazas:

- Alguna empresa con más poder podría copiar la idea y eclipsar a Chikara dejándola en el olvido o como una mera anécdota.
- Al tratarse de una empresa de desarrollo incipiente, el usuario puede desconfiar de nosotros como desarrolladores y de Chikara como producto.



Fortalezas:

- Se trata de una red social que realmente nace con el propósito de aportar valor y lo hace quitando todo lo que sobra, centrándose en un aspecto concreto: la motivación.
- Es una aplicación creada para ser accesible desde cualquier dispositivo o plataforma.
- Permite a los usuarios mantenerse en el anonimato decidiendo si hacer sus publicaciones privadas o no, y utilizarla para el único propósito de inspirarse.

Oportunidades:

- La motivación personal es un tema crucial en la época en la que estamos viviendo. En un mundo diseñado para distraernos y hacernos consumir pasivamente, algo que nos ayuda a enfocarnos y nos ofrece un empujón en un momento de necesidad puede ser un gran aliado.
- Hay personas con más o menos destreza para crear elementos audiovisuales capaces de conmover y llamar a la acción de los demás, pero todos se benefician de un chik de calidad. No hay por qué utilizar exclusivamente elementos propios para que el propósito tenga éxito.

5.2. Innovación

Nuestro principal cometido, a diferencia de otras plataformas es motivar. Por lo tanto, nuestros usuarios tienen el aliciente de encontrar una fuente de motivación cuando acceden a Chikara, y no solo la necesidad de conectar y compartir contenido o consumirlo, aunque también puedan. Chikara es una red social con una función bien diferenciada.

La experiencia del usuario es siempre diferente porque cada persona acude a Chikara con sus propias necesidades. El amplio margen de personalización que eso ofrece nos aporta una experiencia de usuario única y valiosa que puede marcar la diferencia.

PLANIFICACIÓN



6. PLANIFICACIÓN

Las horas efectivas de trabajo para el alcance actual de proyecto han sido definidas por el equipo de desarrollo en base a lo establecido en el boletín oficial del estado (BOE), *“Real Decreto 405/2023, de 29 de mayo, por el que se actualizan los títulos de la formación profesional del sistema educativo de Técnico Superior en Desarrollo de Aplicaciones Multiplataforma y Técnico Superior en Desarrollo de Aplicaciones Web, de la familia profesional Informática y Comunicaciones, y se fijan sus enseñanzas mínimas.”*, el cual establece una duración de 25 horas para el módulo *“0492 - proyecto de desarrollo de aplicaciones multiplataforma”* que se desarrollan en el último trimestre del segundo curso del ciclo formativo.

Dichas horas se dividen en 4 grupos diferenciados con diferente número de horas establecidas, aunque queda a libre asignación por cada desarrollador mientras no exceda un 15% el máximo de horas de cada grupo, a continuación, se detalla dichos máximos por desarrollador.

- Reuniones internas y externas: 20h
- Formación técnica: 30h
- Desarrollo e Investigación: 70h
- Pruebas: 10h
- Documentación: 20h
- Presentaciones: 2h
- **TOTAL: 152h**

Para el control de tareas y horas usamos el software web *“Jira”* de [Atlassian](#), ya que este nos proporciona muchas herramientas para el desarrollo como, por ejemplo, un registro horario por tareas, un cronograma para ver que tareas tienen mas prioridad, o un Kanban para saber en que estado se encuentran.

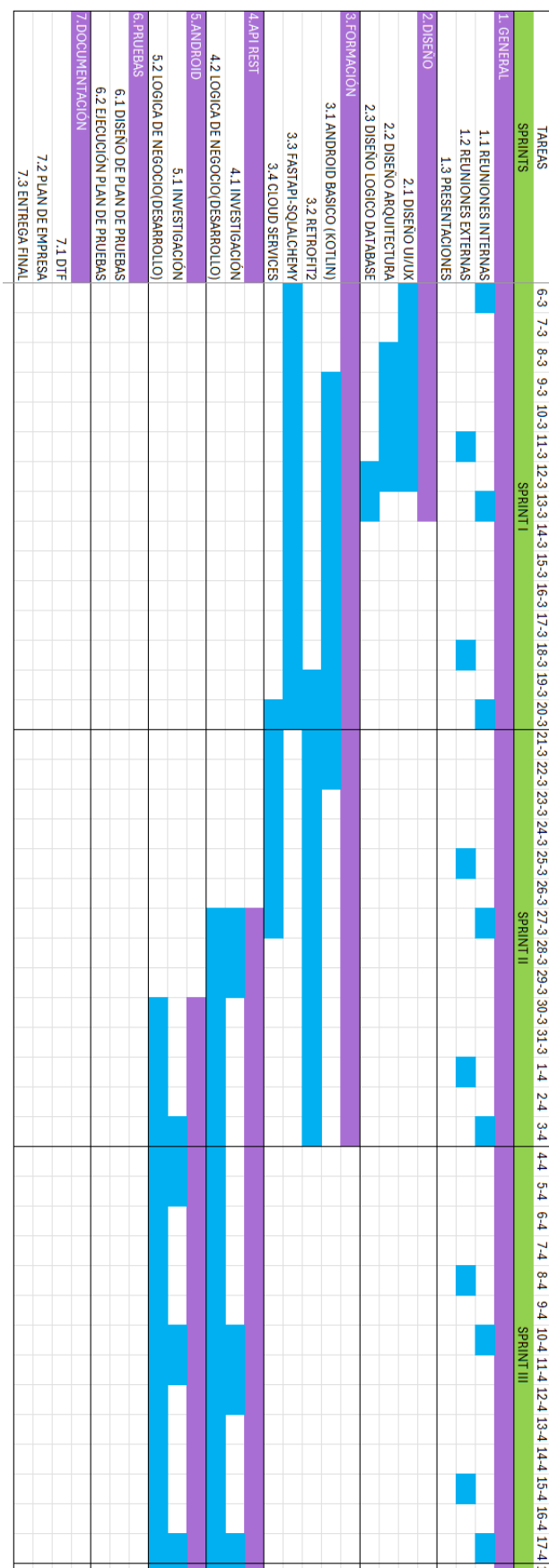
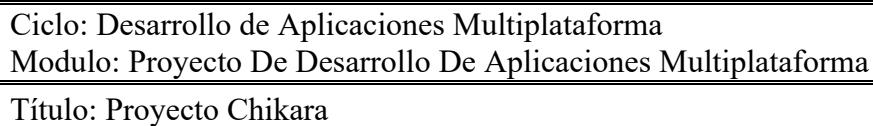


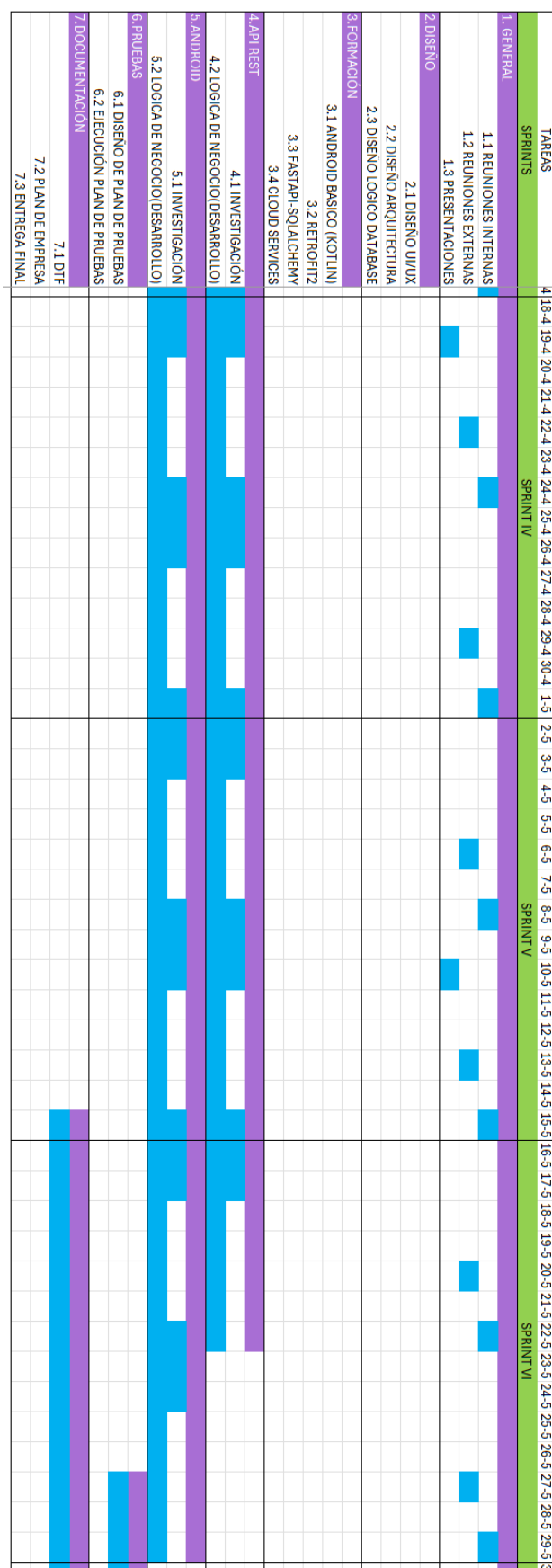
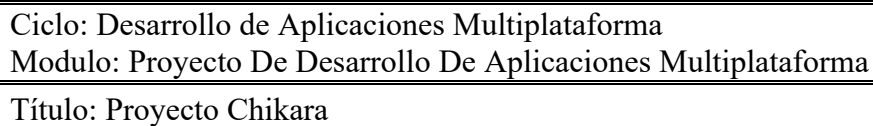
6.1. Diagrama de Gantt

Una vez determinado el número de horas para el alcance actual del proyecto, definimos la distribución en días desde el 06 de marzo de 2024 hasta el 16 de junio de 2024, un total de 102 días, es decir cada desarrollador debe como mínimo emplear 1h y 30min diarios, si no es así, dichas horas deben de ser compensadas en los días posteriores previos al último día del proyecto.

Para ser más eficaces en el desarrollo se utiliza una metodología ágil de trabajo conocida como “*SCRUM*”, por la cual se establecen periodos cortos durante el desarrollo denominados “*SPRINTS*”, en nuestro caso de 2 semanas, al final de los cuales se debe presentar avances en el desarrollo.

A continuación, se muestra un diagrama de Gantt con dichos periodos y lo que se tiene que desarrollar en ellos.







6.2. Recursos técnicos y humanos

Para la fase primera del proyecto se requieren 3 desarrolladores multiplataforma, para todos los aspectos de la fase.

Además, se requiere una cuenta en Azure para el despliegue de los entornos de desarrollo/producción y 3 equipos que cuenten con el siguiente software:

- Interprete de Python 3.11.6 o superior.
- IDE para Python, preferiblemente VSCode o PyCharm.
- Android SDK
- Compilador e intérprete de Kotlin
- Android Studio
- Docker
- GIT
- Jira, puede ser versión web

DISEÑO



7. DISEÑO

El diseño es la fase mas fundamental del desarrollo, ya que lo que se plantea en esta fase condiciona al resto. Nuestro diseño se basa en nuestra competencia con ligeras modificaciones, ya que estos llevan años en el mercado y tienen un equipo muy amplio de diseñadores, por lo que saben lo que mejor funciona en una red social.

Nuestro primer paso fue diseñar un **mockup** y unos casos de uso, para luego determinar toda la **infraestructura** que vamos a necesitar.

7.1.Mockup

Los mockups son una herramienta invaluable en el proceso de diseño, permitiendo una visualización clara y realista de los diseños antes de su implementación final. Al utilizar mockups de manera efectiva, los diseñadores pueden obtener retroalimentación temprana, ahorrar tiempo y recursos, y garantizar la satisfacción del cliente. A continuación, se muestra el mockup de la aplicación móvil de chikara.

La app debe contener una pantalla de carga que sirva para comprobar que el servidor api rest esta disponible, se pasa a la pantalla de login si hay conexión, si no se muestra en mensaje de fallo de conexión, tras pasar el login la navegación múltiple desde un menú inferior nos permite acceder a cualquier información en menos de dos clicks (pulsaciones).

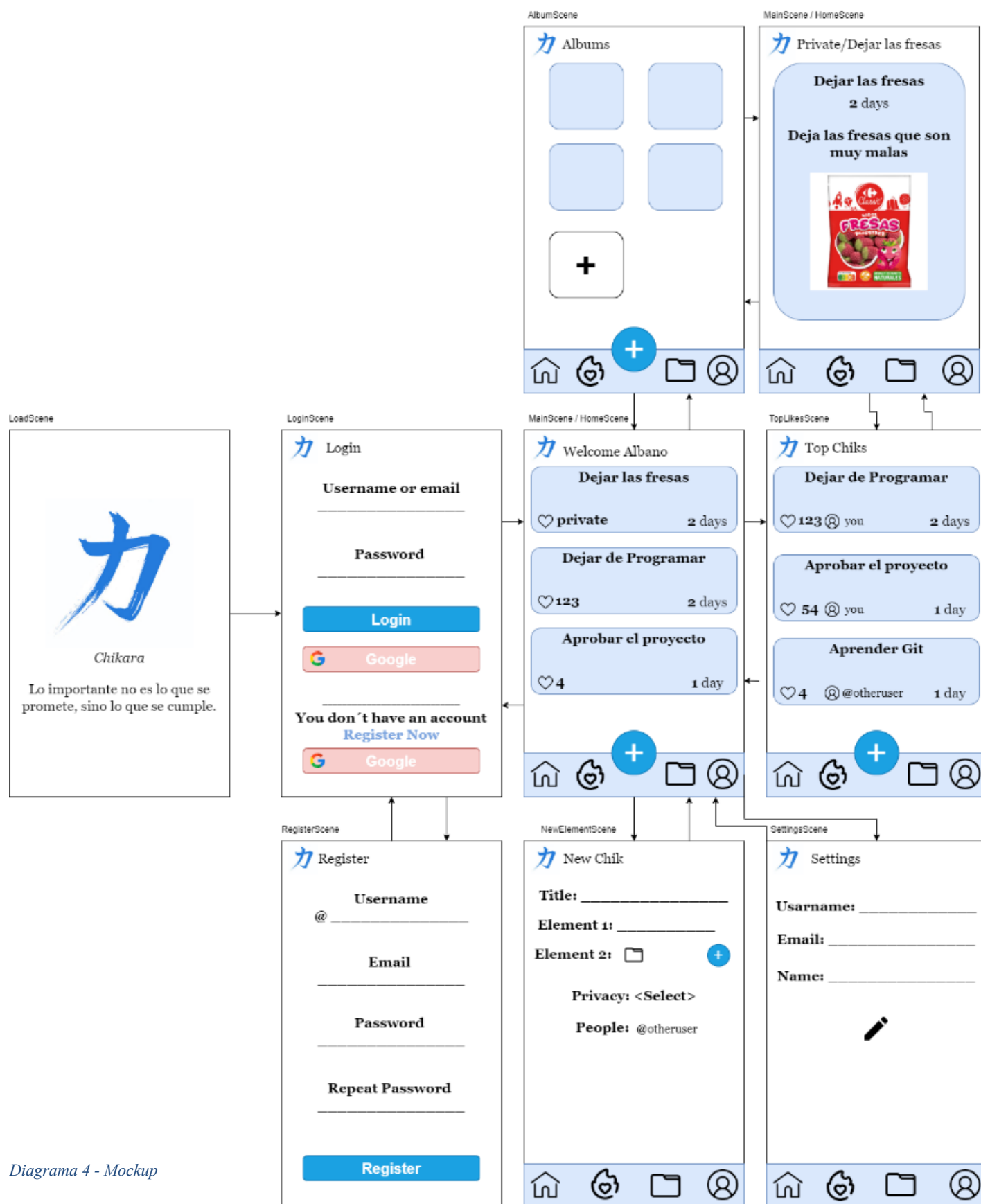


Diagrama 4 - Mockup

7.2.Casos de Uso

Los diagramas de uso son una herramienta esencial en el desarrollo de software, permitiendo una comprensión clara y detallada de cómo interactúan los usuarios con un sistema. Al utilizar diagramas de uso de manera efectiva, los equipos de desarrollo pueden clarificar requisitos, detectar problemas de diseño y diseñar interfaces de usuario intuitivas.

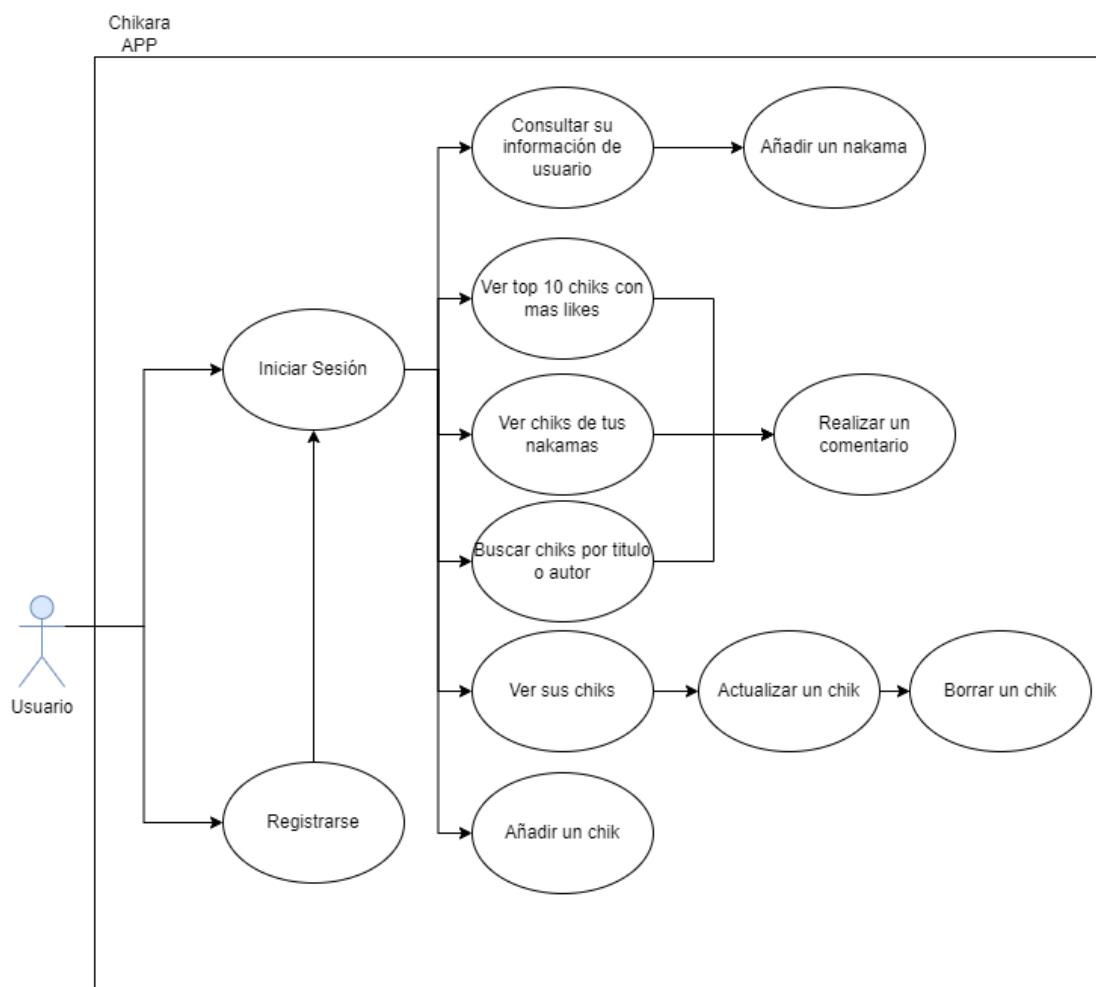


Diagrama 5 - Diagrama de Uso



7.3. Modelos de datos

El modelado de datos es una práctica esencial en el desarrollo de software que permite definir la estructura y las relaciones entre los datos utilizados por una aplicación. Al utilizar modelos de datos de manera efectiva, los equipos de desarrollo pueden clarificar la estructura de los datos, detectar problemas de diseño y crear una base sólida para el desarrollo de la base de datos.

En nuestro caso se optó por una aplicación con dos sistemas de almacenamiento de datos, uno relacional para almacenar la información de los usuarios y otro no-relacional para almacenar la información de los **chiks**, relacionando ambos por la **UUID** de usuario.

7.3.1. PostgreSQL

El primer paso fue decidir qué información del usuario íbamos a guardar, decidimos que íbamos a guardar los datos básicos del usuario, los dispositivos con los que ha iniciado sesión, un registro de inicio de sesiones, y que **nakamas** tiene cada usuario, para almacenar toda esa información diseñamos el siguiente esquema.

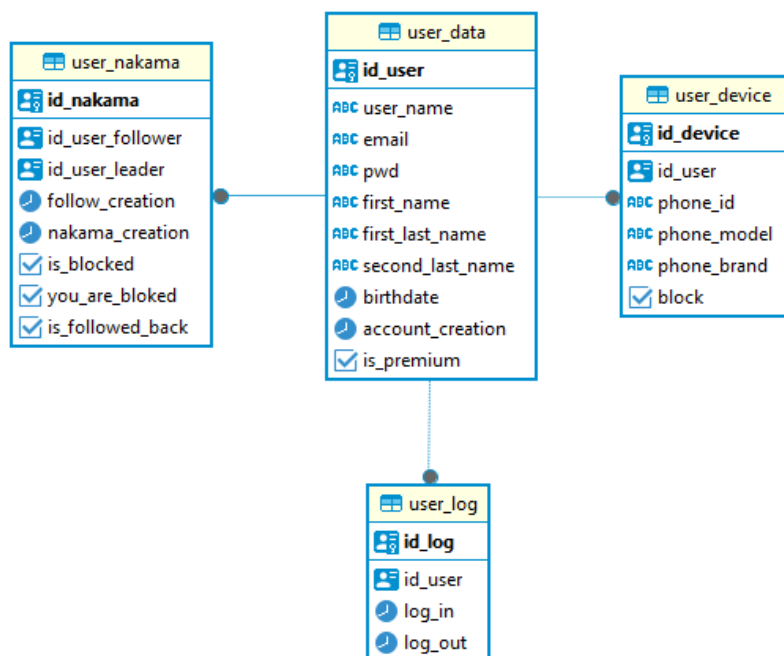


Diagrama 6 - Modelo Relacional



7.3.2. MongoDB

Para el resto de la información que necesita chikara se almacena en documentos de [Mongo](#) con estructura [JSON](#), a continuación, se muestra un ejemplo de [JSON](#) almacenado.

```
1 {
2   "_id": "1cad1ea5-b411-4895-812d-f5c3bf696519",
3   "title": "Prueba Chik",
4   "author": "156844a7-2be5-4208-b46d-7794fe698034",
5   "date": "2024-03-21",
6   "likes": 12,
7   "isprivate": true,
8   "content": [
9     {
10      "position": 1,
11      "value": "Dejar las chuches",
12      "type": "TYPE_TEXT"
13    },
14    {
15      "position": 2,
16      "value": "https://www.giantfreakinrobot.com/wp-content/uploads/2022/06/hellotherethumb.jpg",
17      "type": "TYPE_IMG"
18    }
19  ],
20   "comments": [
21     {
22       "user": "c3b63b33-281c-403f-8129-0ea95e56682f",
23       "comment": "Que chulo",
24       "date": "2024-03-21"
25     },
26     {
27       "user": "c28a49c1-63ae-4944-81d9-ba58f60741fa",
28       "comment": "Que guay",
29       "date": "2024-03-21"
30     }
31   ],
32   "mentions": ["c3b63b33-281c-403f-8129-0ea95e56682f", "c28a49c1-63ae-4944-81d9-ba58f60741fa", "1cad1ea5-b411-4895-812d-f5c3bf696519"]
33 }
```

Código 1 - Chik JSON

7.4. Estructura del proyecto

Dado que el proyecto es multiplataforma, hemos considerado esta característica al plantear su estructura. Hemos decidido desarrollar un proyecto cliente-servidor para lo cual hay que desarrollar una API genérica que pueda ser utilizada por múltiples clientes en diferentes plataformas. A continuación, se presenta un esquema general que se detallará en los apartados [7.4.1 Arquitectura Back-end](#), [7.4.2 Arquitectura Front-end](#) y [7.5 Despliegue](#). Es importante destacar que este no es un esquema UML, ya que la complejidad del proyecto, con su gran número de clases, lo haría difícil de entender.

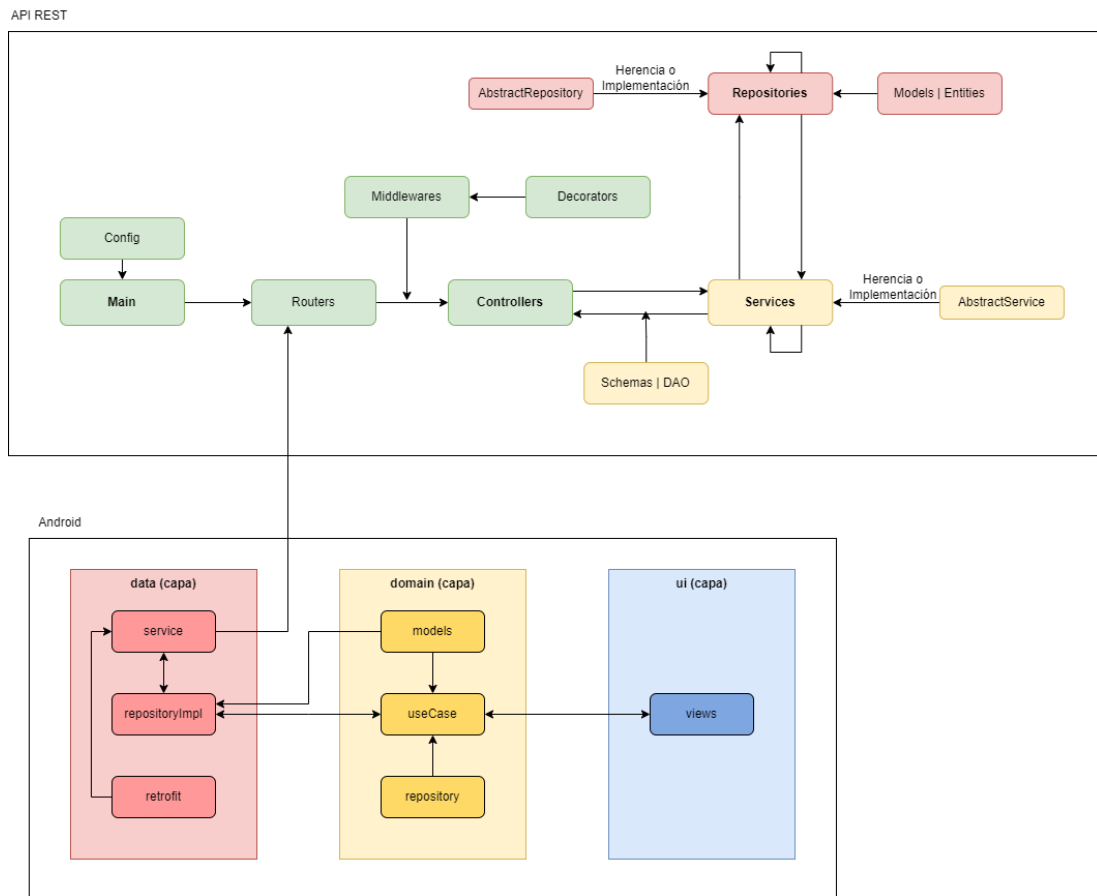


Diagrama 7 - Estructura Chikara

7.4.1. Arquitectura Back-end

El backend está desarrollado con Python, ya que este cuenta con muchas librerías para programar de forma sencilla toda la lógica de negocio. Las principales dependencias son FastAPI, SQLAlchemy, PyMongo y Azure Services. La Arquitectura de clases que hemos seguido es la de Controller-Service-Repository, por la cual, cada elemento tiene una funcionalidad distinta lo cual facilita el mantenimiento y escalabilidad de la API Rest, que se detalla a continuación.

- **MAIN**

- **Propósito:** Punto de inicio del servidor.
- **Responsabilidad:** Configurar y lanzar el servidor. Suele incluir la configuración inicial, como la carga de configuraciones, la conexión a la base de datos y la configuración de rutas y middleware.



- **Ejemplo:** En una aplicación FastAPI, aquí es donde definirías la instancia de FastAPI y ejecutarías el servidor.
- **CONFIG**
 - **Propósito:** Configuración de la aplicación.
 - **Responsabilidad:** Gestionar todas las configuraciones necesarias para la aplicación, como configuraciones de la base de datos, variables de entorno, parámetros de configuración, etc.
 - **Ejemplo:** Archivos como config.py en una aplicación Python, donde se definen variables como DATABASE_URL, SECRET_KEY, etc.
- **DECORATORS**
 - **Propósito:** Funciones adicionales aplicadas a otras funciones o métodos.
 - **Responsabilidad:** Añadir funcionalidad extra a las funciones o métodos, como autenticación, autorización, registro de logs, etc.
 - **Ejemplo:** Decoradores en Python que validan tokens de autenticación antes de permitir el acceso a un endpoint.
- **MIDDLEWARES**
 - **Propósito:** Procesamiento adicional de solicitudes y respuestas.
 - **Responsabilidad:** Interceptar las solicitudes antes de llegar a los controladores y/o interceptar las respuestas antes de enviarlas al cliente. Utilizados para tareas como la gestión de CORS, el manejo de sesiones, el registro de solicitudes, etc.
 - **Ejemplo:** Middleware en FastAPI para gestionar CORS o registrar el tiempo de procesamiento de cada solicitud.
- **ROUTERS**
 - **Propósito:** Definición de rutas de la API.
 - **Responsabilidad:** Encapsular y definir las rutas de la API. Organizar las rutas en diferentes módulos para mantener el código modular y limpio.



- **Ejemplo:** En FastAPI, podrías definir routers en diferentes archivos y luego incluirlos en la aplicación principal.
- **CONTROLLERS**
 - **Propósito:** Controladores de la lógica de negocio.
 - **Responsabilidad:** Gestionar las solicitudes entrantes, llamar a los servicios necesarios y devolver las respuestas adecuadas. Actúan como una capa intermedia entre las rutas y los servicios.
 - **Ejemplo:** En FastAPI, estas serían las clases que reciben solicitudes HTTP, llaman a los servicios y devuelven las respuestas.
- **SERVICES**
 - **Propósito:** Implementación de la lógica de negocio.
 - **Responsabilidad:** Contener la lógica de negocio de la aplicación. Interactuar con los repositorios para obtener, procesar y devolver datos según las necesidades de la aplicación.
 - **Ejemplo:** Métodos que implementan la lógica de negocio, como `create_user`, `add_user`, etc., que interactúan con los repositorios para realizar operaciones en la base de datos.
- **ABSTRACTSERVICE**
 - **Propósito:** Definición de interfaces de servicio.
 - **Responsabilidad:** Definir los métodos básicos deben implementar los services. Garantiza que todas las implementaciones de servicios sigan un conjunto estándar de métodos.
 - **Ejemplo:** En Python, esto podría ser una clase base con métodos abstractos que deben ser implementados por todas las clases de servicios concretas.
- **SCHEMAS**
 - **Propósito:** Definición de esquemas.
 - **Responsabilidad:** Definir la estructura de los datos que se envían y reciben a través de la API.
 - **Ejemplo:** Modelos Pydantic para validar y serializar datos de entrada/salida.



- **REPOSITORIES**

- **Propósito:** Acceso a los datos.
- **Responsabilidad:** Contener la lógica para acceder a los datos almacenados en la base de datos. Realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) y otras consultas específicas.
- **Ejemplo:** Clases que contienen métodos para realizar operaciones en la base de datos, como UserRepository con métodos `get_user_by_id`, `create_user`, etc.

- **ABSTRACTREPOSITORY**

- **Propósito:** Definición de interfaces de repositorio.
- **Responsabilidad:** Definir los métodos que los repositorios deben implementar. Garantiza que todas las implementaciones de repositorios sigan un conjunto estándar de métodos.
- **Ejemplo:** En Python, esto podría ser una clase base con métodos abstractos que deben ser implementados por todos los repositorios concretos.

- **MODELS | ENTITIES**

- **Propósito:** Definición de los modelos de datos.
- **Responsabilidad:** Representar la estructura de las tablas de la base de datos y las entidades del dominio. Definir las propiedades de los datos y sus relaciones.
- **Ejemplo:** Clases SQLAlchemy o Pydantic que definen los modelos de datos, como User, Chik, etc.

- **INTERACCIONES ENTRE COMPONENTES**

- Main inicializa la configuración (Config) y define la instancia de la aplicación.
- Routers manejan las rutas de la API y delegan la lógica a los Controllers.
- Controllers llaman a los Services para realizar la lógica de negocio.
- Services interactúan con los Repositories para realizar operaciones en la base de datos.



- Repositories utilizan Models | Entities para representar y manipular los datos.
- AbstractRepository y AbstractService definen interfaces que aseguran consistencia en las implementaciones de Repositories y Services.

7.4.2. Arquitectura Front-end

El frontend ha sido desarrollado en Kotlin, debido a que es el que mas soporte tiene para los dispositivos Android, además hemos usado múltiples dependencias, la principal es Retrofit que nos permite realizar consultas http a al backend (API rest), para tener una buena arquitectura y que el proyecto sea fácilmente escalable hemos usado las siguientes categorías de clases por capa.

- **CAPA "DATA"**

Esta capa es responsable de manejar los datos de la aplicación, ya sea desde una base de datos local, una API remota, o cualquier otra fuente de datos. Incluye los siguientes componentes:

- **service:** Esta es la interfaz que define las operaciones que se pueden realizar para obtener o enviar datos. Es la capa más externa que interactúa con los repositorios.
- **repositoryImpl:** Implementación del repositorio, que se encarga de interactuar con las fuentes de datos (como bases de datos locales o servicios web). En este caso, usa Retrofit para las llamadas a servicios web.
- **retrofit:** Una biblioteca de cliente HTTP que facilita la comunicación con servicios web RESTful. Es utilizado por repositoryImpl para hacer las peticiones de red.



- **CAPA "DOMAIN"**

Esta capa contiene la lógica de negocio de la aplicación y actúa como un intermediario entre la capa de datos y la capa de UI. Incluye:

- **models:** Define las estructuras de datos que se usan en la aplicación, estos modelos son los que la capa de dominio maneja y manipula.
- **useCase:** Representa un caso de uso específico de la aplicación. Es una clase que encapsula una funcionalidad particular de la aplicación, como "obtener una lista de elementos" o "iniciar sesión".
- **repository:** Esta es la interfaz que define los métodos que el useCase necesita para interactuar con los datos. Esta interfaz es implementada por repositoryImpl en la capa de datos.

- **CAPA UI**

Esta es la capa de presentación, que maneja la interfaz de usuario. Contiene:

- **views:** Los componentes de la interfaz de usuario, como actividades, fragmentos, y vistas personalizadas. Esta capa interactúa con los casos de uso (useCase) para obtener datos y presentarlos al usuario.

7.5.Despliegue

Para el despliegue hemos decidido utilizar Azure debido a su sólida infraestructura en la nube, que ofrece escalabilidad, seguridad y una amplia gama de servicios integrados que pueden satisfacer las necesidades de nuestro proyecto en crecimiento. Azure proporciona soluciones flexibles y eficientes para el almacenamiento de datos y desarrollo de aplicaciones, permitiéndonos innovar rápidamente y optimizar nuestros procesos operativos. Además, la capacidad de Azure para integrarse con nuestras herramientas (VSCode), junto con su compromiso con la protección de datos Europa, asegura que podamos mantener altos estándares de calidad y seguridad en el proyecto.



Hemos optado por implementar una arquitectura de microservicios en Azure. Utilizando servicios como Azure Container Registry, Azure App Web o Azure Blob Storage, podemos desarrollar, desplegar y gestionar componentes independientes que se comunican entre sí de manera eficiente. Esta estrategia nos permite realizar actualizaciones y escalar servicios específicos sin afectar el funcionamiento general del sistema, asegurando un desarrollo más ágil y una mayor resiliencia operativa. A continuación, se muestra un esquema de comunicaciones entre servicios, y una explicación de cada servicio.

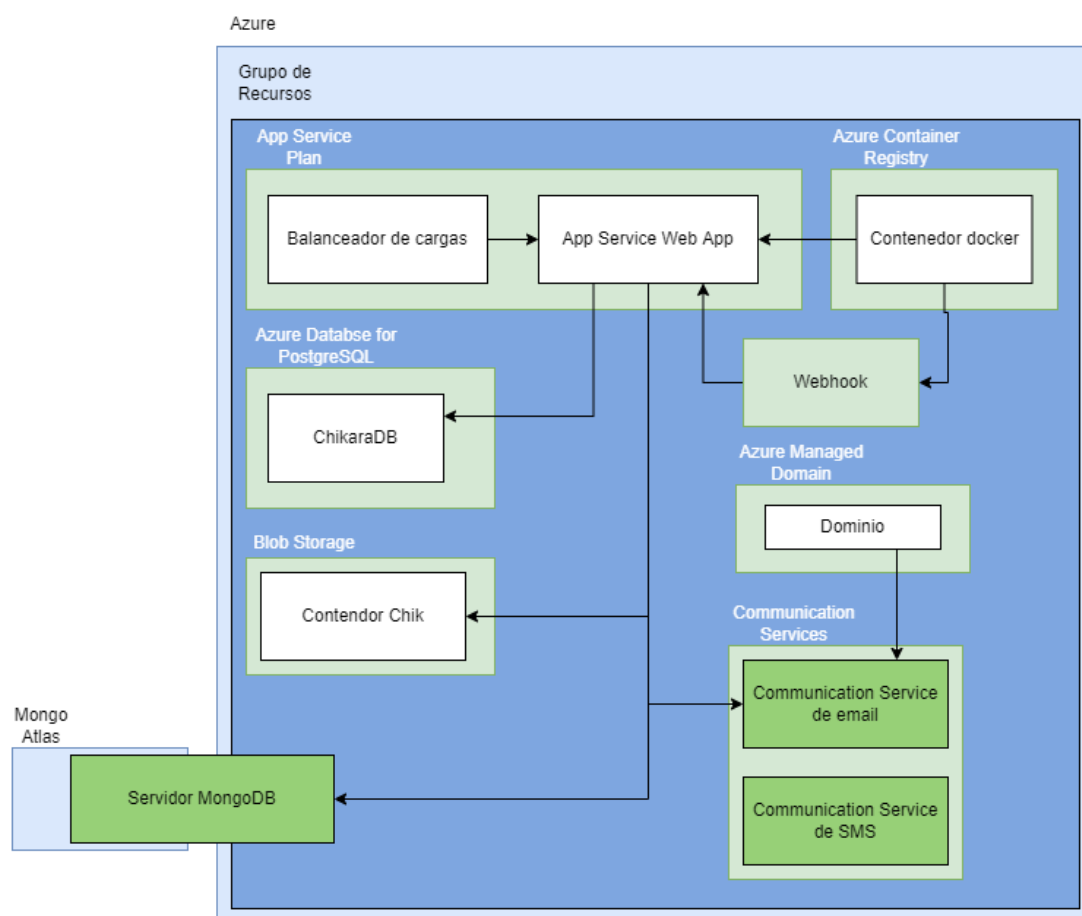


Diagrama 8 - Estructura Azure



1. **Grupo de Recursos (Resource Group):** Es una agrupación lógica de los recursos de Azure utilizados en la solución.
2. **App Service Plan:**
 - **Load Balancer:** Distribuye el tráfico de entrada entre varias instancias de la aplicación web para garantizar alta disponibilidad y escalabilidad.
 - **App Service Web App:** Hospeda y ejecuta la aplicación web.
3. **Azure Container Registry:**
 - **Docker Container:** Repositorio para almacenar y gestionar imágenes de contenedores Docker.
 - **Webhook:** Mecanismo para notificar eventos o actualizaciones desde el contenedor Docker hacia la aplicación web.
4. **Azure Database for PostgreSQL:**
 - **ChikaraDB:** Base de datos PostgreSQL que almacena datos relacionales utilizados por la aplicación web.
5. **Blob Storage:**
 - **Chik Container:** Almacena objetos binarios grandes como archivos, imágenes, videos, etc.
6. **Azure Managed Domain:**
 - **Dominio:** Servicio gestionado que proporciona dominio personalizado para la aplicación.
7. **Communication Services:**
 - **Communication Service de email:** Servicio para enviar correos electrónicos.
 - **Communication Service de SMS:** Servicio para enviar mensajes de texto SMS.
8. **Mongo Atlas:**
 - **Servidor MongoDB:** Base de datos NoSQL en MongoDB Atlas utilizada para el almacenamiento de datos no relacionales y que utiliza los servidores de Azure para funcionar.

IMPLEMENTACIÓN



8. IMPLEMENTACIÓN

Texto

PUESTA EN MARCHA



9. PUESTA EN MARCHA

Texto

PLAN DE PRUEBAS



10. PLAN DE PRUEBAS

Texto

PLAN DE EMPRESA



11. PLAN DE EMPRESA

11.1. IDENTIFICACIÓN

11.1.1. Historia de la empresa

Todo comenzó en la máquina del café.

Gracias a la aleatoriedad con la que ésta suelta los palitos con los que se remueve, comenzamos a hablar entre nosotros. Muy pronto, cuando a Daniel se le ocurrió traer palitos de otra máquina diferente, comenzamos a ayudarnos los unos a otros. Siempre éramos los mismos. Siempre los componentes de la empresa actual.

Nos echábamos un cable en todo lo que podíamos y, sin darnos cuenta, lo que en un principio era una cuestión de compañerismo terminó convirtiéndose en una amistad.

Trabajamos muy bien juntos, porque compartimos muchos valores y, gracias a que los campos de los que procedemos son muy dispares, abarcamos todos los pilares que nuestra empresa necesita para funcionar.

Café con Palito tuvo su estreno triunfal en el segundo reto del primer curso del ciclo formativo DAM del curso 22/23, donde demostró ser una empresa eficaz y competitiva con unas calificaciones impresionantes (y merecidas), por un trabajo del que, aún en el segundo curso, seguimos rescatando código para nuevos trabajos.

11.1.2. Cultura empresarial

Nuestra misión es cubrir las necesidades presentes y futuras, que suponen los avances informáticos y tecnológicos para las pequeñas y medianas empresas, nuestra empresa se enfoca a facilitarle el tránsito aportando soluciones personales a cada una, para adaptarlo rápida y satisfactoriamente a los nuevos tiempos.



Nuestro campo de actividad se centra en el desarrollo de software e implantación de servicios / soluciones informáticas en un mercado local / regional centrado en pequeñas y medianas empresas.

La empresa Café con Palito comienza como una empresa de ámbito regional trabajando de forma local, captando clientes de forma directa, pero con pretensiones de convertirse en nacional, e incluso internacional, en un corto periodo de tiempo ya que el sector tecnológico, específicamente el desarrollo de software y soluciones digitales lo permite con relativa facilidad.

Empezaremos trabajando los 4 fundadores de la empresa y a medida que ganemos clientes, iremos contratando empleados que se alineen con los valores de nuestra empresa.

Nuestros valores los resumimos con una frase. *“Caminando hacia la Agenda 2030”*.

- **Innovación:**

Buscamos constantemente nuevas formas de crear experiencias multimedia únicas y emocionantes para nuestros clientes.

- **Creatividad:**

Fomentamos un entorno donde la creatividad florezca, permitiendo que cada miembro del equipo aporte ideas innovadoras.

- **Calidad:**

Nos comprometemos a ofrecer productos de la más alta calidad, asegurándonos de que cada aplicación multimedia cumpla con los estándares más exigentes.



- **Colaboración:**

Creemos en el poder de la colaboración. Trabajamos estrechamente con nuestros clientes y dentro del equipo para lograr resultados excepcionales.

- **Adaptabilidad:**

Nos adaptamos rápidamente a los cambios tecnológicos y a las necesidades del mercado.

- **Transparencia:**

Fomentamos la honestidad y la transparencia en todas nuestras interacciones, tanto internas como externas.

- **Responsabilidad Social:**

Contribuimos positivamente a la sociedad mediante iniciativas de responsabilidad social empresarial y prácticas éticas.

- **Empatía:**

Nos preocupamos por comprender las necesidades y perspectivas de nuestros clientes, usuarios y colegas, actuando con empatía en todas las situaciones.

- **Inclusividad:**

Celebramos la diversidad y promovemos un ambiente inclusivo donde todas las voces son valoradas y respetadas.

- **Flexibilidad:**

Adoptamos una mentalidad flexible para adaptarnos a cambios rápidos y satisfacer las demandas del mercado y de nuestros clientes.



- **Desarrollo Profesional:**

Apoyamos el crecimiento y desarrollo profesional de nuestro equipo, proporcionando oportunidades de aprendizaje y capacitación continua.

- **Inspiración:**

Buscamos inspirar a través de nuestro trabajo, generando experiencias multimedia que no solo cumplen funciones prácticas, sino que también generan emociones y experiencias memorables.

- **Compromiso con la Calidad de Vida:**

Reconocemos la importancia del equilibrio entre el trabajo y la vida personal, promoviendo un entorno que respeta el bienestar de nuestros empleados.

- **Inversión en la Comunidad:**

Contribuimos al desarrollo de las comunidades en las que operamos, apoyando proyectos locales y participando activamente en iniciativas comunitarias.

- **Agilidad:**

Adoptamos metodologías ágiles para la gestión de proyectos, permitiendo respuestas rápidas a cambios y entregas iterativas eficientes.

- **Pasión por la Tecnología:**

Fomentamos una pasión compartida por la tecnología, inspirando a nuestro equipo a mantenerse actualizado y entusiasmado con las últimas tendencias y avances.



11.1.3. Identificación del equipo

11.1.4. Descripción actividad empresarial

11.2. ANALISIS DE MERCADO

11.2.1. Análisis Macroentorno

11.2.2. Datos generales del sector

11.2.3. Clientes Potenciales

11.2.4. Plan de Marketing

11.2.5. DAFO

11.3. PLAN TECNICO-PRODUCTIVO

11.3.1. Servicios Ofrecidos

11.3.2. Tabla de Precios

11.3.3. Distribución

11.4. PLAN DE RECURSOS HUMANOS

11.4.1. Perfiles profesionales

11.4.2. Organigrama

11.4.3. Política Salarial

11.4.4. Política de Teletrabajo

11.4.5. Política acoso Laboral

11.5. PLAN DE INVERSIONES

11.5.1. Localización Sede

11.5.2. Inmovilizado intangible.

11.5.3. Inmovilizado Material

11.5.4. Presupuesto Detallado de Gastos

11.5.5. Resumen de Inversiones

11.5.6. Fuentes de financiación

11.6. OBLIGACIONES EMPRESARIALES

11.6.1. Forma jurídica

11.6.2. Ventajas



11.6.3. Desventajas

11.6.4. Pasos que rellenar para formar la empresa

11.6.5. Obligaciones Laborales

11.6.6. Obligaciones sobre prevención Laboral

CONCLUSIONES



12. CONCLUSIONES

Texto

BIBLIOGRAFÍA



13. BIBLIOGRAFÍA

Texto

ÍNDICES



14. ÍNDICES

14.1. Índice de Tablas

Texto

14.2. Índice de Ilustraciones

Texto

ANEXOS



ANEXOS

Anexo I Registro Horario

Texto

Anexo II Estatutos de Empresa

Texto

Anexo III Modelo 036

Texto

PREGUNTAS FRECUENTES



PREGUNTAS FRECUENTES

1. ¿Por qué se ha usado un modelo de datos relacional y un modelo de datos no-relacional conjuntamente?

Texto

2. ¿Cómo se relacionan ambos modelos de datos usados?

Texto

3. ¿Por qué hemos usado el servicio cloud Azure de Microsoft sobre otras alternativas?

Texto

4. ¿Por qué usamos una arquitectura de microservicios en vez de una monolítica?

Texto

5. ¿El desarrollo móvil es solo para dispositivos con SO Android, tenemos planes para un desarrollo para SO iOS? ¿Qué lenguaje usaríamos?

Texto

6. ¿Si se deja planteado un desarrollo para múltiples dispositivos porque no hemos usado un framework multiplataforma?

Texto