Universidad CENFOTEC

Proyecto 3

BISOFT-22

Plan General de Brainful

Quantum Tech

Profesores

Prof. Nelson Allan Araya Alvarado

Prof. Olman Josué Santamaría Acosta

Prof. Jessica Cerdas Álvarez

23 de junio de 2024

Audiencia

El presente documento es dirigido hacia todos los integrantes del equipo y a los facilitadores del curso.

Tabla de contenidos

I. Introducción

El presente documento tiene como objetivo proporcionar una guía integral para el desarrollo del software Brainful, una aplicación dedicada a la prevención de la demencia. Este plan general establece los lineamientos y estrategias clave que guiarán las fases de desarrollo, implementación y mantenimiento del sistema, asegurando que se cumplan los estándares de calidad y que se gestionen adecuadamente los riesgos, la configuración, los estándares y el tiempo del proyecto.

II. Definiciones, acrónimos y abreviaturas

- EDT: Es una herramienta de gestión de proyectos que descompone el trabajo total en partes más pequeñas y manejables para facilitar la planificación, ejecución y control.
- SQL: Lenguaje de programación estándar utilizado para gestionar y manipular bases de datos relacionales, permitiendo realizar consultas, inserciones, actualizaciones y eliminaciones de datos.
- Asana: Plataforma de gestión de proyectos y tareas en línea que permite a los equipos planificar, organizar y realizar un seguimiento del trabajo de manera colaborativa y eficiente.
- Github: Plataforma de alojamiento de código fuente que utiliza el sistema de control de versiones Git, facilitando la colaboración, revisión de código y gestión de proyectos de software.
- Angular: Framework de desarrollo web de código abierto, mantenido por Google, que se utiliza para construir aplicaciones web dinámicas y de alta eficiencia con

una arquitectura basada en componentes.

- Repositorio: Espacio centralizado donde se almacenan y gestionan archivos, especialmente código fuente y documentos relacionados, a menudo utilizado en sistemas de control de versiones como Git.
- Stored Procedure: Conjunto de instrucciones SQL que se almacenan en la base de datos y pueden ser ejecutadas como una unidad, mejorando la eficiencia y seguridad de las operaciones repetitivas y complejas.
- VS Code: Editor de código fuente gratuito desarrollado por Microsoft, conocido por su flexibilidad, soporte para múltiples lenguajes de programación y extensiones que amplían sus funcionalidades.

III. Plan de administración de la calidad

i. Formato y detalle de las Listas de Verificación

Las listas de verificación van a servir para asegurar la calidad de los documentos y entregables que se van a ir a desarrollando a lo largo del cuatrimestre para el proyecto de Brainful. En dichas listas se va a poder observar;

1. **Descripción de los elementos a revisar**: un breve resumen del elemento a ser evaluado y sus criterios de evaluación.
2. **Cumple/ No cumple:** Una indicación a observar si el segmento del documento cumple o no con los requisitos.
3. **Fecha de revisión:** La fecha que se realizó la entrega a revisión.
4. **Encargado de revisión:** Nombre del encargado de revisar el apartado.
5. **Observaciones:** Observaciones que se notan en el proyecto

ii. Diseño y especificación de los casos de prueba (pruebas unitarias)

Las pruebas unitarias se realizarán en la herramienta de jUnit, estas pruebas se integrarán durante la fase de desarrollo del software y se van a ejecutar antes de cada revisión de sprint para asegurar la funcionalidad de los módulos.

jUnit funciona de forma que hace pruebas automatizadas para asegurar que el código en java funcione correctamente, utilizamos para que haga pruebas diseñadas para verificar el funcionamiento de unidades individuales del código.

La verificación será responsabilidad del desarrollador, pero los resultados serán verificados por el líder de equipo en cada revisión del sprint.

Los casos de prueba se elaborarán para validar el comportamiento individual de cada parte del sistema de brainful, los casos de prueba estarán enfocados en verificar una funcionalidad específica o un escenario particular, esto incluye la preparación de los casos de prueba y la revisión de los mismos donde se compararán con los resultados esperados.

El formato aplicado para los casos de prueba será, deberá tener un nombre, pasos para ser reproducido, secciones de evidencia, todo esto lo desarrollaremos en el Azure DevOps con su contenido específico para estos tipos de pruebas, además de necesitar tener la fecha de cuando se realizó la prueba.

iii. Diseño y especificación de las pruebas de rendimiento

Las pruebas de rendimiento de Brainful, las utilizaremos en jMeter, estas pruebas se realizan para evaluar el comportamiento de la aplicación bajo diferentes condiciones y cargas de trabajo, ya que es una aplicación muy competente y ampliamente utilizada para pruebas de rendimiento, permitiendo simular cargas pesadas y analizar el comportamiento de la aplicación. Se van a hacer una serie de pruebas antes de cada iteración para tener un control general de la aplicación.

Una pequeña configuración para jMeter podría darse por pasos, como por ejemplo, creación del Test plan, agregar un Thread group, que es donde se

define el número de usuarios virtuales y la duración de la prueba, configuración de las http request, esto lo que hace es simular solicitudes http que realizan los usuarios virtuales, además de especificar el método si es (get, post, delete), esos serían los pasos para una pequeña configuración de jMeter.

Ahora una pequeña explicación de las pruebas de rendimiento que tiene jMeter, tiene validación de plan de pruebas que lo que hace es verificar las configuraciones de los Thread Group, Samplers, Assertions, para que estén correctamente configurados, además de tener una ejecución de pruebas, donde iniciamos la ejecución del test plan anteriormente creado y así poder observar y registrar los resultados de las pruebas que se están ejecutando, análisis de resultados, jMeter cuenta con listeners para analizar las métricas claves de rendimiento, como tiempos de respuesta, errores o rendimiento de la CPU y por último la generación de informes que trae consigo jMeter, donde generamos informes detallados donde vienen gráficos de comportamiento y la estabilidad de la aplicación.

Pantalla que simula la prueba de rendimiento de jMeter

iv. Calendarización

La calendarización describe las actividades de calidad a realizar durante el proyecto de Brainful

Actividad	Semana
Creación de listas de verificación	6-8
Revisión de listas de verificación	6-8
Desarrollo de pruebas unitarias	7-10
Ejecución de pruebas unitarias	10
Revisión de documentación de las pruebas	11
Pruebas de rendimiento (jMeter)	12
Ajuste de pruebas	13-14
Entrega final	15

IV. Plan de Gestión de riesgos

i. Tabla de riesgos

Posibles riesgos			
Riesgo	Descripción	Mitigación	Contingencia
Desarrollo			
Falta de cooperación en el equipo	Los miembros del equipo no colaboran efectivamente, afectando la productividad y el progreso del proyecto.	Reuniones semanales de equipo sobre trabajo. Designar roles claros y responsabilidades.	Reasignar tareas críticas, realizar reuniones de mediación, en caso de no servir el coordinador deberá definir si las personas pueden seguir trabajando juntas o deben de separarse.
Documentación incompleta	La documentación técnica y de usuario no está completa, lo que dificulta la comprensión y mantenimiento del sistema.	Establecer revisiones regulares de la documentación. Asignar responsables específicos para la documentación.	Realizar una auditoría completa de la documentación y reasignar tareas para completarla rápidamente.
Estimaciones de tiempo incorrectas	Las tareas toman más tiempo del estimado, lo que retrasa el cronograma del proyecto.	Utilizar metodologías ágiles para revisiones y ajustes continuos. Realizar estimaciones con margen de error.	Redefinir el cronograma, reasignar recursos adicionales o solicitar horas extras temporales.
Entrega de tareas tarde	Las entregas de componentes del sistema no cumplen con las fechas establecidas, afectando las siguientes fases del proyecto.	Establecer plazos intermedios y revisiones periódicas. Proporcionar soporte adicional si es necesario.	Priorizar tareas, ajustar el cronograma global, y comunicar los cambios a todas las partes asignadas.

Curva de aprendizaje muy grande	Los miembros del equipo tienen dificultades para adaptarse a las nuevas tecnologías, lo que afecta la eficiencia.	Proporcionar capacitación continua y recursos de aprendizaje entre los mismos compañeros. Consultas técnicas a profesores en caso de ocuparse.	Reasignación temporal de tareas, reuniones de equipo para capacitación, en caso de no poderse o todos los miembros ser afectados solicitar a profesor reunión.
Producto			
Juegos no efectivos	Los juegos y ejercicios cognitivos no logran conectar en los usuarios o no se ajustan adecuadamente a las necesidades individuales de los usuarios.	Realizar pruebas de usuario tempranas y frecuentes. Ajustar el diseño basado en la retroalimentación.	Revisión y rediseño de juegos basado en experiencias encontradas de su investigación inmediata.
Problemas de usabilidad	La interfaz del usuario no es intuitiva ni accesible, lo que provoca una mala experiencia de usuario.	Realizar pruebas de usabilidad durante el desarrollo. Incluir criterios de accesibilidad desde el inicio.	Implementar mejoras de usabilidad basadas en pruebas adicionales y retroalimentación.
Falta de retroalimentación adecuada	El sistema no proporciona suficiente retroalimentación a los usuarios sobre su progreso y áreas de mejora.	Diseñar mecanismos de retroalimentación desde el inicio. Probar con usuarios finales para asegurar efectividad.	Actualizar el sistema para incluir retroalimentación, realizar iteraciones rápidas de mejora.
Configuración			
Configuración incorrecta	Errores en la configuración inicial de la aplicación que afectan su funcionamiento.	Crear guías de configuración detalladas y realizar pruebas de configuración.	Revisar y corregir la configuración, aviso a compañeros de errores y en caso de ocuparse reunión de emergencia en búsqueda de solución.

Problemas de integración	Dificultades para integrar los módulos de ejercicios y desafíos con otras partes de la aplicación.	Planificar la integración desde las primeras fases del desarrollo. Realizar pruebas de integración continua.	Reasignar recursos para solucionar problemas de integración, buscar soluciones temporales como llamado de página sin el procedimiento planteado.
Calidad			
Pruebas insuficientes	No se realizan pruebas de calidad suficientes, lo que resulta en errores y fallos en la aplicación.	Establecer un plan de pruebas exhaustivo. Incluir pruebas automatizadas y manuales.	Implementar una fase de pruebas intensiva, el equipo completo deberá de verse involucrado.
Falta de retroalimentación de pruebas	No se recibe suficiente retroalimentación durante las pruebas, lo que dificulta la identificación de problemas.	Involucrar a usuarios finales desde fases tempranas. Crear canales de retroalimentación eficientes.	Ampliar el periodo de pruebas, recopilar y analizar más datos de uso real para identificar problemas.
Fallos en el control de calidad	Los procedimientos de control de calidad no detectan errores Importantes antes de la presentación.	Realizar auditorías de calidad internas basadas en información previamente verificada de internet y pruebas extra.	Revisar y reforzar los procedimientos de control de calidad.

ii. Tabla de impactos

Tabla de Evaluación de Impactos		
Impacto	Descripción	Gravedad
E	Consecuencias capaces de borrar parte del trabajo o el deploy.	5
D	Errores que evitan la continuación práctica del trabajo.	4
C	Errores que comprometen funcionalidades del proyecto.	3
B	Errores que no afectan la funcionalidad principal del proyecto.	2
A	Errores ortográficos.	1

Impacto de riesgos	
Riesgo	Impacto en Proyecto
Desarrollo	
Falta de cooperación en el equipo	D
Documentación incompleta	A
Estimaciones de tiempo incorrectas	C
Entrega de tareas tarde	C
Curva de aprendizaje muy grande	C
Producto	

Juegos no efectivos	C
Problemas de usabilidad	D
Falta de retroalimentación adecuada	C
Configuración	
Configuración incorrecta	D
Problemas de integración	D
Calidad	
Pruebas insuficientes	C
Falta de retroalimentación de pruebas	C
Fallos en el control de calidad	C

iii. Análisis de probabilidad

Matriz de probabilidad		
Ocurrencia	Significado	Valor
Frecuente	Casi certeza que se produzca.	5
Probable	Probable que se produzca.	4
Ocasional	Probable que se produzca a veces.	3
Posible	Puede ocurrir en algún momento.	2
Improbable	Nunca puede ocurrir.	1

Probabilidad de riesgos	
Riesgo	Probabilidad de Ocurrencia
Desarrollo	
Falta de cooperación en el equipo	Improbable
Documentación incompleta	Probable
Estimaciones de tiempo incorrectas	Ocasional
Entrega de tareas tarde	Ocasional
Curva de aprendizaje muy grande	Ocasional
Producto	
Juegos no efectivos	Posible
Problemas de usabilidad	Improbable
Falta de retroalimentación adecuada	Posible
Configuración	
Configuración incorrecta	Posible
Problemas de integración	Posible
Calidad	
Pruebas insuficientes	Posible
Falta de retroalimentación de pruebas	Posible
Fallos en el control de calidad	Posible

iv. Análisis de probabilidad e impacto.

Rangos de Riesgo	
Descripción	Gravedad
Alto	10 +
Medio	6-9
Bajo	1-5

Riesgos			
Riesgo	Impacto	Nivel de Ocurrencia	Rango de Riesgo
Desarrollo			
Falta de cooperación en el equipo	4	1	Bajo
Documentación incompleta	1	4	Bajo
Estimaciones de tiempo incorrectas	3	3	Medio
Entrega de tareas tarde	3	3	Medio

Curva de aprendizaje muy grande	3	3	Medio
Juegos no efectivos	3	2	Medio
Problemas de usabilidad	4	1	Bajo
Falta de retroalimentación adecuada	3	2	Medio
Configuración incorrecta	4	2	Medio
Problemas de integración	4	2	Medio
Pruebas insuficientes	3	2	Medio
Falta de retroalimentación de pruebas	3	2	Medio
Fallos en el control de calidad	3	2	Medio

V. Plan de administración de la configuración

i. Descripción de las herramientas

Herramienta de Control de Versiones: Git

Descripción: Git es una herramienta de control de versiones que permite a los desarrolladores gestionar y almacenar sus cambios en el código fuente de manera eficiente y segura.

Proveedor del Servicio: Git es una herramienta de código abierto mantenida por la comunidad de desarrolladores, con servicios populares como GitHub, GitLab y Bitbucket que ofrecen soluciones basadas en la nube.

Uso en el Proyecto: En el proyecto, Git se utilizará para el seguimiento y control de versiones de todo el software. Esto incluye el almacenamiento de código y cualquier otro archivo relevante, que esté dentro del ámbito codificado del proyecto.

Plataforma de Gestión de Repositorios: GitHub

Descripción: GitHub es una plataforma de alojamiento de repositorios Git basada en la nube, que proporciona una interfaz amigable, herramientas de colaboración, integración continua, y gestión de proyectos.

Proveedor del Servicio: GitHub Inc., una subsidiaria de Microsoft.

Uso en el Proyecto: GitHub será utilizado para alojar los repositorios de proyectos, facilitando la colaboración entre los integrantes del equipo de desarrollo, la revisión de código, y la gestión de issues y pull requests.

Herramienta de Gestión de Proyectos y Backlog: Azure DevOps

Descripción: Azure DevOps es un conjunto de herramientas de desarrollo en la nube que ofrece servicios de gestión de proyectos, control de versiones, integración continua y entrega continua (CI/CD). Incluye Azure Boards para la gestión de backlog, Azure Repos para repositorios Git, Azure Pipelines para CI/CD, Azure Test Plans para pruebas, y Azure Artifacts para gestión de paquetes.

Proveedor del Servicio: Microsoft, conocido por su software y servicios en la nube bajo la marca Azure.

Uso en el Proyecto: Se utiliza para toda la gestión del backlog del proyecto, ahí mismo se utiliza su sistema de épicas, product backlog y tareas. Para facilitar el manejo y el progreso de los integrantes del equipo.

Sistema de Respaldo: Google Drive

Descripción: Google Drive es un servicio de almacenamiento en la nube que permite a los usuarios almacenar archivos y sincronizarlos entre dispositivos. Ofrece herramientas de colaboración en tiempo real a través de Google Docs, Sheets y Slides.

Proveedor del Servicio: Google LLC, que ofrece una amplia gama de servicios

en línea y productos de hardware.

Uso en el Proyecto: En un drive se guardan todos los documentos utilizados para el desarrollo del proyecto, además de tener los mockups del proyecto y los diagramas de entidades del proyecto de programación.

ii. Administración del repositorio

Estructura de repositorio backend

gradle/wrapper: Contiene archivos necesarios para utilizar el Gradle Wrapper, facilitando la configuración del entorno de construcción.

src: Contiene el código fuente del proyecto, con subcarpetas para el código principal y las pruebas.

.gitignore: Define qué archivos y directorios deben ser ignorados por Git.

LICENSE: Define los términos bajo los cuales se distribuye el proyecto.

Procfile: Configuración específica para el despliegue de la aplicación.

README.md: Proporciona una visión general del proyecto, cómo configurarlo y utilizarlo.

build.gradle: Script de construcción de Gradle para el proyecto.

gradlew y gradlew.bat: Scripts para ejecutar Gradle Wrapper en Unix y Windows, respectivamente.

settings.gradle: Archivo de configuración de Gradle.

system.properties: Configuraciones adicionales del sistema.

Estructura de repositorio frontend

src: Contiene el código fuente de la aplicación Angular, incluyendo componentes, servicios, módulos, recursos estáticos y configuraciones de entorno.

.**DS_Store:** Archivo del sistema macOS que debería ser ignorado por Git.

.gitignore: Define qué archivos y directorios deben ser ignorados por Git.

LICENSE: Define los términos bajo los cuales se distribuye el proyecto.

README.md: Proporciona una visión general del proyecto, cómo configurarlo y utilizarlo.

angular.json: Archivo de configuración para el proyecto Angular, define la estructura del proyecto y cómo se construye.

package-lock.json: Archivo de bloqueo de versiones para asegurar que las dependencias instaladas sean exactamente las mismas en todos los entornos.

package.json: Contiene las dependencias del proyecto y los scripts para la construcción, pruebas y despliegue.

tsconfig.app.json: Configuración de TypeScript específica para la aplicación.

tsconfig.json: Configuración de TypeScript general para el proyecto.

tsconfig.spec.json: Configuración de TypeScript específica para pruebas.

Responsables del repositorio

El encargado del repositorio será todo el equipo; sin embargo, en la rama de QA el encargado solo será el coordinador de calidad del proyecto, entre las reponsabilidades se pueden mencionar las siguientes:

- Asegurar que la estructura del repositorio se mantenga ordenada y actualizada.
- Revisar y aprobar las pull requests.
- Gestionar los permisos de acceso al repositorio.

Flujo de Trabajo

Para el desarrollo del proyecto, se utilizará un flujo de trabajo donde cada integrante tendrá su propia rama con su nombre. Esto permitirá a cada miembro trabajar de manera independiente antes de integrar sus cambios en una rama común que en este caso será QA y finalmente de QA se enviará a producción una vez se hayan hecho las pruebas de calidad respectivas. El flujo de trabajo es el siguiente:

Rama Principal (main/master): Contiene el código estable y listo para producción.

Rama de Desarrollo de cada integrante (nombre del integrante): Contiene el código más reciente y las nuevas funcionalidades en desarrollo.

Rama de QA (QA): Contiene el código que debe ser probado por QA, es a la

rama que se envían las nuevas funcionalidades.

Rama de Producción (Produccion): Contiene el código que ya fue probado por calidad y está actualmente desplegado.

iii. Reglas del repositorio

Desarrollo y Commits

Los cambios se desarrollan en la rama del integrante.

Se realizan commits frecuentemente con mensajes claros y descriptivos, para saber en qué se está trabajando y tener trazabilidad de los cambios en cada rama.

Pull Requests y Revisión de Código

Una vez que una funcionalidad está completa, el integrante crea una pull request (PR) desde su rama hacia QA, para que esta sea aprobada por calidad y le den el visto bueno para ser enviada a main.

La PR debe ser revisada y aprobada por al menos otro miembro del equipo y debe tener una descripción clara de lo que se va a integrar con el merge.

Integración en QA

Después de la aprobación del PR, se fusiona la rama del integrante con QA, para que se hagan las pruebas de calidad, se realizan las respectivas pruebas

de calidad en esta rama y se da el visto bueno.

Integración en main

Periódicamente, las ramas de QA se integran en main para crear versiones estables y finalmente se envían a Producción para desplegarse.

VI. Plan de estándares

i. Estándares de interfaz gráfica

Los colores escogidos para Brainful fueron un color celeste (#CAF0F8) el cual se utilizara para los fondos y áreas grandes debido a que este es el color dominante de la aplicación. Este color se escogió, ya que el color azul suele hacer referencia al ámbito de la salud. Lego como color secundario se utilizará un turquesa (#16C2D5) el cual se utilizara para secciones de contenido, fondos de tarjetas, y otros elementos secundarios. Por último, el color de acentos será el naranja (#FF9F1C) el cual se utilizara para botones, enlaces, y otros elementos interactivos. También se utilizarán diferentes tonalidades de estos colores para dar contraste a cada uno de estos colores en donde se necesiten. Se Utilizara la regla de 60-30-10 donde el 60% es el color dominante, 30% es el color secundario y el 10% son los colores de acento.

Para la nomenclatura de los controles se nombran usando una abreviatura del control seguido de su funcionalidad por ejemplo si se utiliza un botón para el inicio de sesión entonces se deberá llamar btn-inicioSesion, en caso de un checkbox se utilizará check, y así con cada control que se vaya a utilizar. Se

emplea el uso de Bootstrap para la creación eficiente de componentes y controles dentro de la aplicación.

Se utilizarán iconos fáciles de identificar para ciertas acciones de la aplicación para facilitar la experiencia de los usuarios y no aglomerar la aplicación con mucho texto.

ii. Estándares de codificación

El uso de comentarios a través de todo el proyecto será de suma importancia para aclarar ciertos bloques de códigos que se hagan muy extensos.

Estándares de la base de datos

Se planea usar una nomenclatura clara, consistente y significativa a través de las tablas, columnas, constraints y store procedures. Se utilizará un guion bajo para separar palabras en los nombres para una mejor legibilidad. En caso de utilizar stored procedures se nombran usando las primeras tres letras de la acción seguido de su función, por ejemplo CRE_USER_SP. Utilizar la indentación adecuada para los queries y además usar mayúsculas para las palabras reservadas de SQL. Hacer uso de comentarios en caso de que el query sea muy amplio y pueda crear confusión a los demás integrantes del equipo.

Estándares de código en angular

Para la nomenclatura de los componentes, servicios directivos y entre otros se utilizará la guía de nomenclatura oficial de Angular para tener una mayor consistencia en general a través del proyecto. Se agruparán los archivos por

funcionalidad para tener una mejor organización de cada componente que está relacionado a dicha función. De esta manera se tendrá una mejor modularidad, escalabilidad y flexibilidad en la estructura de la aplicación.

Estándares de Javascript

Se utilizará el camelCase para el nombramiento de funciones y variables y el PascalCase para el nombramiento de clases. Las variables y funciones deberán de tener un nombre significativo a su función para que sea fácil de entender. Se exigirá el uso correcto de indentación, puntos y comas y corchetes para tener una mejora, organización y legibilidad del código. Si es posible, se usarán extensiones de VS Code que ayudarán con la identificación y formato del código.

Estándares de Java

Se utilizará el camelCase para el nombramiento de funciones y variables y el PascalCase para el nombramiento de clases. Las variables y funciones deberán de tener un nombre significativo a su función para que sea fácil de entender. Sé exigirá el uso correcto de indentación, puntos y comas y corchetes para tener una mejora, organización y legibilidad del código. En el caso de Spring boot el nombramiento de los endpoints también deberán ser significativos a su funcionalidad y clase a la que se esté dirigiendo.

iii. Estándares de documentación

En esta sección se debe incluir los estándares de tamaño de letra de cada título y de cada subtítulo, así como del texto de los documentos.

Para la documentación del equipo de trabajo, los estándares serán los siguientes:

1. Encabezados de Comentario
    - Título (Negrita): 16pt
    - Subtítulo (Negrita): 13pt
2. Texto del Cuerpo
    - Texto Normal:12 pt
3. Fuente:Dosis

Directrices

- Concisión: Mantén los comentarios en línea breves y al punto, utilizando un tamaño de letra más pequeño para evitar sobrecargar el código.
- Claridad: Utiliza negritas o cursivas de manera moderada para diferenciar encabezados y subtítulos del texto principal del comentario.
- Consistencia: Asegúrate de que los tamaños de letra y estilos se mantengan consistentes en todos los comentarios en línea dentro del código o la

documentación.

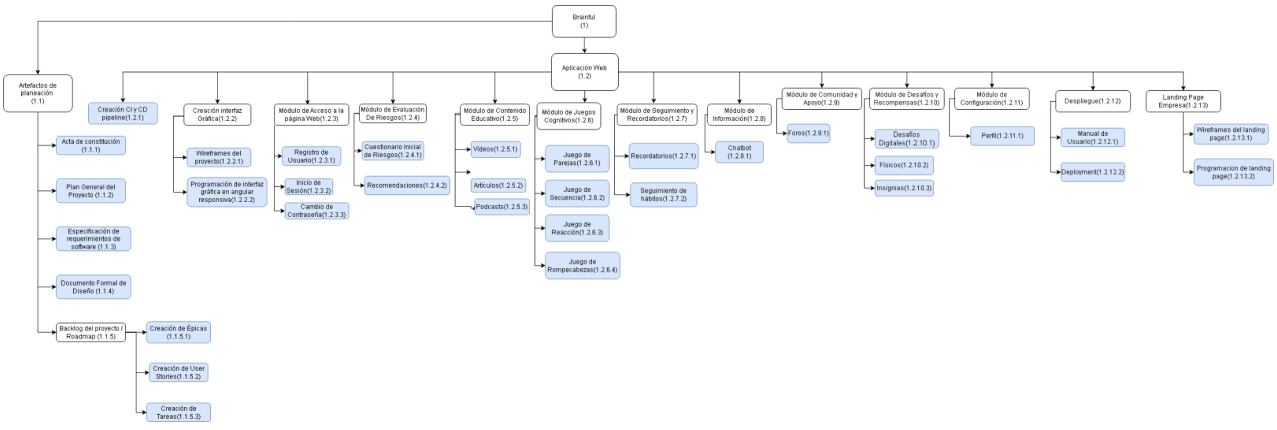
Hoja de Figma del Proyecto

https://www.figma.com/design/NCc5mxOg0CNajeeNibwLIl/Barinful?node-id=0-1&t=Bs675AH6K4vFLAfu-1

VII. Plan de administración del tiempo

En el desarrollo del proyecto se van a utilizar ciertas herramientas para llevar el control de las tareas , la administración del proyecto y su propia documentación. En primer lugar , se va a utilizar una carpeta de Google Drive en la cual se van a almacenar los artefactos de documentación y demás elementos relacionados con el proyecto. En segundo lugar, para la gestión del proyecto y el control de versiones vamos a utilizar github y github desktop. Por último, para la creación de tareas se va a utilizar Azure devops.

i. Estructura de desglose de trabajo (EDT)



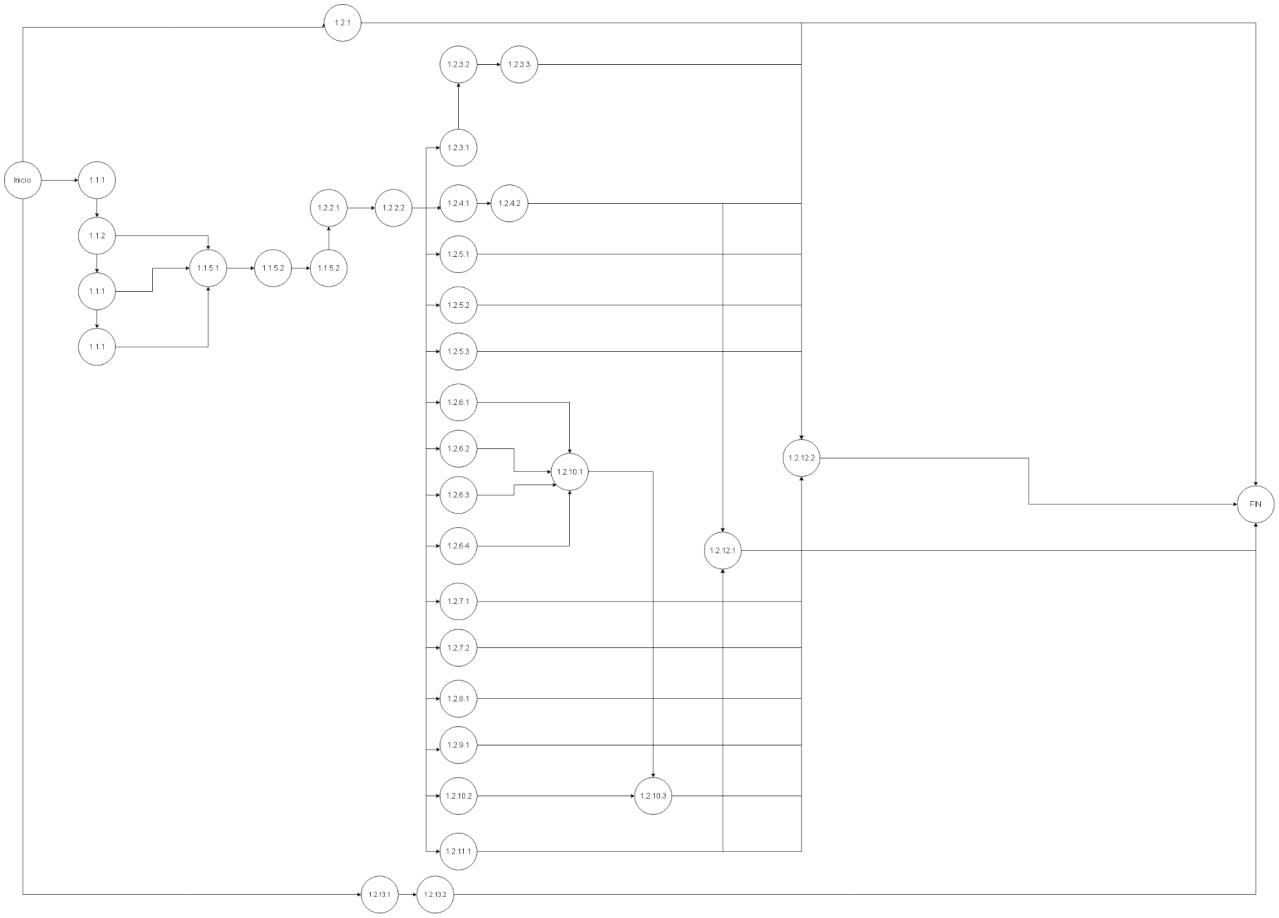
https://drive.google.com/file/d/1VdKIgi38doPA1yM2QdmxfCfDR7Pekf-x/view?usp=sharing

ii. Tabla de dependencia de actividades

ID	Actividad	Dependencias
1.1.1	Acta de constitución	-
1.1.2	Plan General del Proyecto	1.1.1
1.1.3	Especificación de requerimientos de software	1.1.2
1.1.4	Documento Formal de Diseño	1.1.3
1.1.5.1	Creación de Épicas	1.1.2,1.1.3,1.1.4
1.1.5.2	Creación de User Stories	1.1.5.1
1.1.5.3	Creación de Tareas	1.1.5.2
1.2.2.1	Wireframes del proyecto	1.1.5.3
1.2.2.2	Programación de interfaz gráfica en angular responsiva	1.2.2.1
1.2.3.1	Registro de Usuario	1.2.2.2
1.2.3.2	Inicio de Sesión	1.2.3.1
1.2.3.3	Cambio de Contraseña	1.2.3.2
1.2.4.1	Cuestionario Inicial de Riesgos	1.2.2.2
1.2.4.2	Recomendaciones	1.2.4.1
1.2.5.1	Vídeos	1.2.2.2
1.2.5.2	Artículos	1.2.2.2
1.2.5.3	Podcasts	1.2.2.2
1.2.6.1	Juego de Parejas	1.2.2.2
1.2.6.2	Juego de Secuencia	1.2.2.2

1.2.6.3	Juego de Reacción	1.2.2.2
1.2.6.4	Juego de Rompecabezas	1.2.2.2
1.2.7.1	Recordatorios	1.2.2.2
1.2.7.2	Seguimiento de hábitos	1.2.2.2
1.2.8.1	Chatbot	1.2.2.2
1.2.9.1	Foros	1.2.2.2
1.2.10.1	Desafíos Digitales	1.2.6.1,1.2.6.2,1.2.6.3,1.2.6.4
1.2.10.2	Físicos	1.2.2.2
1.2.10.3	Insignias	1.2.10.1,1.2.10.2
1.2.11.1	Perfil	1.2.2.2
1.2.12.1	Manual de Usuario	1.2.3.1, 1.2.3.2, 1.2.3.3, 1.2.4.1, 1.2.4.2, 1.2.5.1, 1.2.5.2, 1.2.5.3, 1.2.6.1, 1.2.6.2, 1.2.6.3, 1.2.6.4, 1.2.7.1, 1.2.7.2, 1.2.8.1, 1.2.9.1, 1.2.10.1, 1.2.10.2, 1.2.10.3, 1.2.11.1, 1.2.12.1
1.2.12.2	Deployment	1.2.1,1.2.3.1, 1.2.3.2, 1.2.3.3, 1.2.4.1, 1.2.4.2, 1.2.5.1, 1.2.5.2, 1.2.5.3, 1.2.6.1, 1.2.6.2, 1.2.6.3, 1.2.6.4, 1.2.7.1, 1.2.7.2, 1.2.8.1, 1.2.9.1, 1.2.10.1, 1.2.10.2, 1.2.10.3, 1.2.11.1, 1.2.12.1
1.2.1	Creación CI y CD pipeline	-
1.3.1.1	Solución de Problemas	1.2.12.2
1.3.1.2	Gestión de riesgos	1.2.12.2
1.3.2.1	Implementación de actualizaciones basadas en petición de usuarios	1.2.12.2
1.3.2.2	Revisión del rendimiento de software	1.2.12.2
1.2.13.1	Wireframes del landing page	
1.2.13.2	Programación de landing page	1.2.13.1

iii. Diagrama de Red



https://drive.google.com/file/d/1tFN4903qusuaEpL3i81kP6hPxvr5TULy/view?usp=sharing