

**INGENIERÍA EN DESARROLLO Y GESTIÓN DE SOFTWARE**

**ASIGNATURA: GESTIÓN DEL PROCESO DE DESARROLLO DE SOFTWARE**

AUTORES:

ANGEL JOSUE HERNANDEZ MANUEL 20210672

AGUSTINA HERNANDEZ MARTINEZ 20200700

JUAN CARLOS CIFUENTES CRUZ 20210654

**GRADO:** 10° **GRUPO:** B

**DOCENTE: ING. ANA MARIA FELIPE REDONDO**

HUEJUTLA DE REYES, HIDALGO. 04 DE OCTUBRE DE 2024

1. **Planeación del desarrollo orientado a la App Móvil**
2. **Objetivos y Alcance del proyecto** 
   1. **Objetivos**

Desarrollar una aplicación móvil para proporcionar a los usuarios una plataforma integral que les permita, a través de una cuenta personal, contratar servicios de ambulancia de manera rápida y eficiente, gestionar el registro de citas médicas, y realizar donaciones para apoyar causas humanitarias, todo en un entorno accesible, seguro y conveniente. Estas funcionalidades mejorarán la interacción de los usuarios con los servicios esenciales y facilitarán un canal para apoyar causas importantes para la empresa Cruz Roja.

* 1. **Alcance**

El alcance de la aplicación móvil incluye el desarrollo de una plataforma que permita a los usuarios, mediante la creación de una cuenta, acceder a diversas funcionalidades clave: la contratación de servicios de ambulancia en tiempo real, el registro y gestión de citas médicas, y la posibilidad de realizar donaciones a causas relacionadas con servicios de salud. La aplicación integrará sistemas de notificaciones para recordar citas y confirmar servicios. Además, incluirá un sistema de perfiles de usuario donde podrán consultar el historial de servicios contratados, citas agendadas.

1. **Equipo de trabajo y roles ´**

El equipo de trabajo está conformado por los siguientes integrantes:

Ángel Josué Hernández Manuel – Líder del proyecto, Desarrollador y Documentador

Agustina Hernández Martínez – Líder de proyecto, Desarrollador y Documentador

Juan Carlos Cifuentes Cruz – Líder de proyecto, Desarrollador y Documentador

1. **Plazos y entregas**

Para los plazos entregables en la planeación del proyecto, se tomo en cuenta el mes de septiembre a noviembre, bajo el periodo de 4 semanas que contempla cada mes, ya que se trabaja bajo la metodología ágil Scrum, todas los entregables fueron planificadas mediante un diagrama de Gantt que se muestra a continuación.



1. **Riesgos identificados y medidas para mitigarlos**
   1. **Definición de los objetivos y alcances del proyecto**

No tener claridad en la definición de los objetivos y en el alcance del proyecto, lo que podrá ocasionar malos entendimiento dentro del equipo de trabajo y una mala ejecución de las actividades planificadas.

* **Medida de mitigación**:
  + Realizar reuniones con todo el equipo involucrado para definir y acordar los objetivos específicos, generales y el alcance del proyecto.
  + Crear un documento de definición de objetivos y alcances que sea accesible y aprobado por todos los miembros clave del proyecto. Este documento debe incluir metas claras, los entregables esperados y criterios de éxito.
  + Implementar un proceso formal de control de cambios para cualquier ajuste en los objetivos o alcances. Los cambios deben ser documentados y aprobados por las partes interesadas clave.
  1. **Retrasos de entregables**

Retrasos en la entrega de las actividades asignadas en el cronograma de Gantt, debido de tareas complejas.

* **Medida de mitigación:** 
  + Dividir las tareas complejas en subtareas más pequeñas y manejables, lo que permite identificar posibles cuellos de botella y asignar recursos de manera más efectiva.
  + Asegurarse de que las tareas más complejas sean asignadas a los miembros del equipo con mayor experiencia y conocimientos técnicos en esas áreas.
  + Establecer revisiones frecuentes del avance de las tareas técnicas, para detectar cualquier retraso temprano y ajustarlo.
  1. **Falta de comunicación**

Falta de comunicación efectiva entre los miembros del equipo, lo que puede llevar a malentendidos con el desarrollo de la aplicación.

* **Medidas de mitigación:**
  + Implementar una matriz de roles y responsabilidades para que todos los miembros del equipo sepan quién es responsable de qué, a quién deben acudir para resolver dudas y quién debe ser informado de los avances o problemas.
  + Definir las herramientas oficiales de comunicación para garantizar que todos los miembros del equipo utilicen los mismos canales.
  1. **Rendimiento y escalabilidad**

La aplicación móvil puede no ser capaz de manejar un aumento en la cantidad de usuarios o en la carga de trabajo, lo que provocaría una disminución del rendimiento, tiempos de respuesta lentos.

* **Medidas de mitigación:** 
  + Realizar pruebas de rendimiento, identificar y corregir cuellos de botella, optimizando los recursos.
  + Implementar una arquitectura basada en microservicios y balanceo de carga para distribuir las solicitudes eficientemente.
  1. **Protección y seguridad de datos**

Vulnerabilidad en la protección y seguridad de los datos puede provocar la exposición de información confidencial, pérdida de datos o incumplimiento de normativas de privacidad, lo que puede llevar a sanciones legales y pérdida de confianza de los usuarios.

* **Medidas de mitigación:** 
  + Implementar cifrado tanto en tránsito (SSL/TLS) como en reposo (AES) para proteger la información sensible.
  + Utilizar autenticación multifactor (MFA) y un sistema de control de acceso basado en roles (RBAC) para limitar el acceso a datos sensibles.
  1. **Integración con el Sitio Web**

Integración inadecuada entre la aplicación móvil y el sitio web puede llevar a incoherencias en la información, problemas de sincronización de datos.

* **Medidas de mitigación:** 
  + Realizar pruebas automatizadas y de compatibilidad para asegurar que la aplicación y el sitio web funcionen correctamente juntos.
  + Implementar herramientas de monitoreo y mantener logs de errores para detectar y corregir fallos rápidamente.
  1. **Experiencia de usuario**

Mala experiencia de usuario puede causar que los usuarios abandonen la aplicación, reduzcan el uso o generen malas críticas, afectando el éxito del producto.

* **Medidas de mitigación:**
  + Realizar pruebas con usuarios reales y crear prototipos para validar la experiencia de usuario antes de la implementación final.
  + Implementar un diseño accesible y responsive, asegurando que sea intuitivo y fácil de usar en cualquier dispositivo.
  + Reducir los tiempos de carga y usar animaciones para mejorar la interacción del usuario.

1. **Estrategias de comunicación**

* Canales de comunicación: Sesión en meet para la comunicación diaria y a discusión rápida de problemas. Establecer un proceso claro para la resolución de conflictos. Por ejemplo, en caso de un malentendido técnico, se podrían organizar sesiones de aclaración en las que todos los miembros participen y expongan sus puntos de vista para llegar a una solución.
* Utilizar la herramienta de Trelloo cualquier herramienta de gestión de tareas para asignar, priorizar y hacer el seguimiento del estado de cada tarea.
* Utilizar un sistema de control de versiones como **Git** con ramas específicas para las plataformas si se requiere personalización por plataforma. Asegurarse de que las integraciones se realicen cuidadosamente y sean revisadas por miembros clave del equipo.

1. **Metodología Ágil de desarrollo**

La metodología ágil seleccionada fue Scrum, el cual es una metodología para la gestión de proyectos en donde nos permitirá obtener un resultado rápido en entornos muy cambiantes, mantiene un marco de trabajo ágil a través del cual, cada integrante del equipo podrá abordar problemas complejos adaptativos y entregar los productos de forma eficiente y creativa, de igual manera esta metodología permite el trabajo colaborativo. La metodología comprende de 4 fases, el cual en cada fase se realizan diferentes funciones para terminar con éxito cada fase.

* **Planificación:** En esta fase el equipo establecerá las tareas prioritarias del proyecto a desarrollar para la aplicación móvil como los objetivos y la lista de lo mas importante para el Product Backlog (Sprint). Estableciendo las prioridades según el valor de cada entregable.
* **Ejecución del Sprint:** En esta fase el equipo trabajara en los elementos seleccionados del Sprint. Desarrollar los diseños, las funcionalidades enfocándonos en cumplir los objetivos cada uno de los Sprint de acuerdo a la planificación inicial.
* **Revisión del Sprint:** Al finalizar las actividades establecidas en el Sprint, se realizan la revisión del trabajo completado. Se verificará si los elementos cumplen con los criterios de aceptación.
* **Retrospectiva del Sprint:** Después de la revisión, se analiza el Sprint completado para identificar áreas de mejoras en el desarrollo o que es lo que funciono durante el Sprint que no.

Entre otros aspectos que se consideraron para la elección de la metodología fueron los siguientes:

* Mantiene un enfoque que se basa en la colaboración y la comunicación constante entre todos los miembros del equipo. Todos deben estar alineados con los objetivos del sprint y del proyecto en general.
* Scrum divide el trabajo en ciclos cortos de desarrollo llamados **sprints**, que normalmente duran de 1 a 4 semanas. Cada sprint termina con un incremento del producto.
* Scrum permite una gran flexibilidad para adaptarse a los cambios de requisitos, prioridades o entornos de negocio durante el desarrollo del proyecto.
* Al final de cada sprint, el equipo realiza una **Sprint Review** y una **Sprint Retrospective**. En la revisión, el equipo muestra el trabajo terminado y recibe feedback. En la retrospectiva, se evalúan los procesos internos para mejorar continuamente.
* Está diseñado para gestionar proyectos complejos y de alta incertidumbre, donde los requisitos pueden cambiar frecuentemente o no están completamente definidos desde el inicio.

1. **Pila tecnológica**

Para el desarrollo de la aplicación se elegio trabajar con las siguientes herramientas para cada capa, como se muestra en la siguiente tabla. Utilizando la pila MERN STACK.

|  |  |  |
| --- | --- | --- |
| Capa de Presentación | Capa de negocio | Capa de Datos |
| Desarrollo en React Native - Devtop  The first alpha of Tailwind CSS v3 is now released - Laravel News  Zustand: Better than React's Context | by Ali Husen | JavaScript in Plain  English | Node.js 18 actualiza el motor de JavaScript y permite crear compilaciones  personalizadas - MuyLinux  Best Practices for Organizing Your Express.js Project: A Guide to Folder  Structure | by Vikas Mishra | FAUN — Developer Community 🐾  Nodemailer – Send e-mails with Node.JS  Como iniciar sesión con Google Api (oAuth 2.0) en ASP.Net 4.7  Company Enrichment API - Abstract API | SQL Server icon PNG and SVG Vector Free Download    What Is Vercel & How Is It For Front-End Developers? |

1. **Issue Tracking y Asignación de Issue**
   1. **Herramienta de planeación**

La herramienta elegida para la asignación de actividades en la planificación es ClickUp, en la cual se integraron los miembros del equipo, así como los docentes a revisar los entregables.

1. **Selección de herramienta de control de versiones**

Para el manejo de versionamiento se eligió GitHub.

Consideramos que dentro del desarrollo del proyecto el control de versiones es esencial para asegurar el trabajo colaborativo y poder trabajar en diferentes funcionalidades sin riesgo de sobrescribir el trabajo de otros. Esta herramienta nos permitirá registrar y revisar cada cambio hecho en el código, con la posibilidad de volver a versiones anteriores si surge algún problema. Esto proporciona un entorno más seguro y organizado para el desarrollo. Al desarrollar la aplicación, los integrantes del equipo podremos trabajar en distintas características o resolver bugs de manera simultánea sin afectar el código principal (branch main o master). Los **pull requests** permiten revisar el código de manera colaborativa antes de integrarlo al proyecto principal, lo que ayuda a garantizar la calidad y a mantener la coherencia en el estilo de codificación. GitHub se integra fácilmente con plataformas de **Integración Continua y Entrega Continua (CI/CD). E**stas integraciones permiten ejecutar automáticamente pruebas unitarias, de integración y de interfaz de usuario cada vez que se realiza un cambio en el código. Además, GitHub permite automatizar el proceso de generación. GitHub permite una gestión avanzada de ramas, lo que facilita a los equipos trabajar en diferentes funcionalidades, mejoras o correcciones de errores sin afectar el código principal hasta que todo esté probado y listo para ser fusionado. GitHub ofrece una forma organizada de gestionar estas ramas, permitiendo que los desarrolladores trabajen en distintas características o correcciones de forma simultánea sin interferencias, hasta que los cambios estén listos para ser integrados en la rama principal. Facilita la automatizacióndepruebas **y** despliegue, lo cual es crucial para asegurar la calidad continua en un entorno de desarrollo rápido como el de React Native.

GitHub permite la colaboración y revisión del código entre los miembros del equipo y asi poder realizar cambios, corrección de errores y agregar modificaciones en caso de que sea necesario para el proyecto.

1. **Selección de estrategia de versionamiento**

Para el desarrollo de nuestra aplicación móvil, hemos optado por utilizar un flujo de trabajo centralizado para el control de versiones. Esta estrategia se adapta perfectamente a las necesidades de un equipo pequeño como el nuestro, compuesto por tres integrantes.

El flujo de trabajo centralizado se basa en el uso de un único repositorio central donde la rama principal (main) contiene siempre la versión estable del código. Todos los desarrolladores realizan cambios directamente en esta rama, asegurando que:

* El código siempre esté listo para ser desplegado en producción.
* La simplicidad del flujo de trabajo facilita la colaboración directa entre los miembros del equipo, sin la necesidad de gestionar múltiples ramas o flujos complejos.

Además, esta estrategia nos permite mantener un control claro sobre los cambios en el proyecto, evitando la complejidad que puede surgir con el uso de múltiples ramas. Como herramienta de control de versiones, utilizaremos GitHub, el cual ofrece una plataforma robusta para la colaboración en tiempo real, la revisión de código y la integración continua (CI) mediante flujos de trabajo automatizados para ejecutar pruebas y asegurar la calidad del código.

1. **Creación, revisión y fusión de ramas.**

Por el tipo de estrategia de versionamiento seleccionado el equipo de desarrollo trabaja única rama central del repositorio, generalmente llamada **"main"** o **"master"**, y las ramas de características o desarrollo son creadas y fusionadas de manera controlada para asegurar la estabilidad del proyecto.

* 1. Creación de la rama

Cuando un miembro del equipo necesita trabajar en una nueva funcionalidad o corrección de errores, se debe crear una nueva rama basada en la rama central (main/master) para aislar los cambios. Para ello se debe asegurar de estar en la última versión de la rama central (main o master) y de que el código esté actualizado.

**git checkout main**

**git pull origin main**

* **Crear una nueva rama**: Crear una nueva rama para la tarea específica. El nombre de la rama debe ser descriptivo y reflejar la tarea que se esté realizando.

git checkout -b feature/nueva-funcionalidad

* Realizar todos los cambios necesarios en la nueva rama, asegurándose de que no afecten la estabilidad del proyecto en la rama principal. A medida del avance, se realizan **commits** regulares con mensajes claros y descriptivos sobre los cambios realizados.

git add .

git commit -m "Descripción clara del cambio realizado"

* **Revisión de la rama:**

Antes de fusionar los cambios en la rama central, la rama debe ser revisada para asegurar que cumple con los requisitos del proyecto y no introduce errores. Una vez que hayas terminado de trabajar en la nueva funcionalidad, sube tu rama al repositorio remoto para que otros puedan revisarla.

git push origin feature/nueva-funcionalidad

**Abrir una solicitud de revisión (Pull Request)**: En la plataforma de control de versiones que es GitHub se crea una **Pull Request (PR)** desde la rama nueva hacia la rama central (main o master).

* **Fusión de ramas:**

Una vez que la revisión ha sido aprobada y no hay más correcciones pendientes, se fusiona la rama al repositorio central. Antes de fusionar, asegurarse de estar en la última versión de la rama central y luego fusiona tu rama.

git checkout main

git pull origin main

git merge feature/nueva-funcionalidad

* 1. **Estrategia contempla la integración continua CI**

Este tipo de estrategia si contempla la integración continua, ya que una estrategia de versionamiento centralizado, la rama principal (normalmente main o master) es el único lugar donde se realiza la integración continua de código. Los desarrolladores crean ramas temporales para desarrollar nuevas funcionalidades o correcciones, pero todos los cambios se integran de nuevo en la rama principal una vez revisados. La Integración Continua en el contexto del versionamiento centralizado se centra en garantizar que la **rama principal siempre esté en un estado estable.** Esto significa que cada vez que se fusionan ramas de características o correcciones a la rama principal, se deben realizar pruebas automatizadas para asegurarse de que la aplicación no se rompa. Para garantizar la calidad del código que se fusiona en la rama central, es crucial utilizar un flujo de **Pull Requests** (PR) donde otros miembros del equipo revisan el código antes de ser fusionado.

1. **Estrategia de despliegue**

Para el despliegue de la aplicación móvil, implementaremos la estrategia Canary, con el objetivo de reducir riesgos al momento de lanzar nuevas versiones del software. Con esta estrategia, se despliega la nueva versión a un subconjunto limitado de usuarios antes de hacerlo disponible para todos. Esto nos permitirá:

* **Minimizar riesgos**: La versión nueva es probada por un pequeño grupo de usuarios, lo que nos permite identificar posibles errores sin afectar a toda la base de usuarios.
* **Recoger feedback temprano**: El comportamiento de la nueva versión puede monitorearse en tiempo real, permitiendo ajustes rápidos y efectivos.
* **Rollback fácil**: Si se detectan problemas, se puede revertir a la versión anterior de manera sencilla, minimizando el impacto en la experiencia del usuario.

Esta estrategia es especialmente útil en proyectos de aplicaciones móviles, donde la experiencia del usuario es crítica. Requiere un monitoreo continuo del rendimiento y el comportamiento del sistema, para lo cual utilizaremos herramientas de monitoreo como Prometheus y Grafana si es necesario a medida que el proyecto crezca. Con el despliegue Canary, garantizamos que el servicio siga funcionando correctamente, mientras introducimos mejoras y nuevas funcionalidades de manera controlada.

1. **Entornos de desarrollo**

Para las herramientas seleccionadas en el entorno de desarrollo son las siguientes:

* **Planificación**

Se eligió la herramienta de ClickUp usado para la gestión de las actividades definidas para el desarrollo del proyecto en donde nos permite organizar y planificar el trabajo del equipo de desarrollo. De igual manera nos permite crear y asignar tareas, fijar fechas límite y realizar un seguimiento del progreso del proyecto. Algunas de sus funcionalidades clave incluyen.

* **Control de versiones:**

Para el control de versiones se eligió trabajar con la herramienta de Git + GitHub.

El cual, con la ayuda del alojamiento de nuestro repositorio en GitHub, nos permite llevar un seguimiento sobre los cambios, correcciones de la aplicación desarrollada y de esta manera poder gestionar las revisiones mediante el pull request y las ramas.

* + **Entorno de desarrollo:**

Para la herramienta del entorno del desarrollo se trabajará en Visual Studio Code.

Es un editor de código muy popular, especialmente para lenguajes como JavaScript y TypeScript, que son esenciales en tecnologías como React Native (para desarrollo móvil) y Node.js (para backend). Algunas de las razones por las que fue elegida la herramienta son las siguientes:

* + - **Extensiones**: Permite añadir extensiones para mejorar la productividad, como soporte adicional para **JavaScript/TypeScript**, **linting**, depuración de código y herramientas de control de versiones como **Git**.
    - **Depuración integrada**: VSCode permite depurar el código directamente desde el editor, lo que facilita identificar y corregir errores durante el desarrollo.
    - **Soporte para React Native y Node.js**: Ofrece un excelente soporte para estas tecnologías, proporcionando una integración fluida con frameworks y bibliotecas usadas en el desarrollo móvil y backend.

**Staging**

**Expo:** Esta herramienta nos facilita el desarrollo y prueba de aplicaciones móviles en React Native. En esta fase de **staging** (preproducción), Expo se utiliza para probar la aplicación en dispositivos reales o simuladores, permitiendo a los desarrolladores ver cómo se comporta la app en un entorno similar al de producción.

**Producción**

Para la parte de producción la herramienta seleccionada es Hostinger.

El cual esta herramienta es un proveedor de servicios de hosting que nos permitirá desplegar el **backend** en un entorno de producción. En esta fase, el backend está disponible para usuarios reales y es accesible en internet.

1. **Como clonar un repositorio**

Para poder realizar la clonación del repositorio lo primero que se debe contemplar es tener instalado la herramienta de Git. Una vez que se tenga la herramienta instalada, se debe abrir la línea se comando o consola de nuestro equipo de computo y ejecutar el siguiente comando:

git clone (URL\_DEL\_REPOSITORIO)

Y posteriormente navegar hacia el directorio para verificar que el proyecto se haya creado.

1. **Instalar dependencias**

Para poder realizar la instalación de las dependencias debemos navegar hacia el directorio en donde se encuentra el proyecto, desde la terminal y una vez ahí adentro ejecutamos el siguiente comando:

npm install

1. **Ejecutar el proyecto**

Para la ejecución del proyecto, de igual manera estando en la terminar se ejecuta el comando:

expo start

Esto abrir o bueno nos proporcionara un código QR. Y escanear ese código para poder abrir la aplicación, un punto muy importante, debemos de tener una aplicación instalada en nuestro aparato móvil que se llama Expo Go.



  