MIGRACIÓN Y ACTUALIZACIÓN DE APLICATIVO WEB Y MÓVIL PARA EL TALLER DE SENSIBILIZACIÓN FRENTE AL CAMBIO CLIMÁTICO

ROBINSON ALBEIRO TIJO GONZALEZ
JAEN FELLINI ECHAVARRIA TIJO

UNIDAD CENTRAL DEL VALLE DEL CAUCA
FACULTAD DE INGENIERÍA
INGENIERÍA DE SISTEMAS
TULUÁ - 2023

MIGRACIÓN Y ACTUALIZACIÓN DE APLICATIVO WEB Y MÓVIL PARA EL TALLER DE SENSIBILIZACIÓN FRENTE AL CAMBIO CLIMÁTICO

ROBINSON ALBEIRO TIJO GONZALEZ
JAEN FELLINI ECHAVARRIA TIJO

PROPUESTA DE TRABAJO DE GRADO PRESENTADO COMO REQUISITO PARA OPTAR AL TÍTULO DE INGENIERO DE SISTEMAS

Directora
MARY LUZ OJEDA SOLARTE
Ingeniera de sistemas

Co director
JOSE GABRIEL PÉREZ CANENCIO
Ingeniero de sistemas

UNIDAD CENTRAL DEL VALLE DEL CAUCA
FACULTAD DE INGENIERÍA
INGENIERÍA DE SISTEMAS
TULUÁ - 2023

NOTA DE ACEPTACIÓN

_____
_____
_____
_____
_____
_____

_____

Presidente del jurado

_____

Jurado

_____

Jurado

Tuluá, 2023

DEDICATORIA

A mis padres, por su amor incondicional, apoyo constante y por ser mi fuente inagotable de inspiración. A mis amigos y seres queridos, por su aliento y compañía en cada paso de este emocionante viaje académico. A mis profesores y mentores, por su sabiduría, orientación y paciencia en mi búsqueda del conocimiento. Esta tesis está dedicada a todos aquellos que han sido parte fundamental de mi camino hacia el logro de este objetivo. Su influencia ha dejado una huella imborrable en mi vida y en este trabajo.

Robinson Albeiro Tijo Gonzalez

Dedico este trabajo a mi familia, cuyo apoyo inquebrantable y confianza han sido el motor que me impulsó a alcanzar este logro. A mis profesores, cuyas enseñanzas, consejos y dedicación han sido indispensables para mi desarrollo académico y personal. A mis amigos y compañeros de clase, quienes compartieron risas, desafíos y conocimientos a lo largo de esta travesía. Esta tesis es un testimonio de la dedicación y esfuerzo colectivo de todos aquellos que han influido en mi camino académico. Que este trabajo contribuya al avance de la tecnología y al mejoramiento de la sociedad.

Jaen Fellini Echavarria Tijo

AGRADECIMIENTOS

Agradezco sincera y profundamente a la profesora Mary Luz Ojeda Solarte y al profesor José Gabriel Pérez Canencio. Sus orientaciones, dedicación y apoyo han sido como faros en este viaje académico. Sus enseñanzas han moldeado mi camino y su aliento me ha dado fuerzas para superar desafíos.

También deseo reconocer a la Universidad Unidad Central del Valle del Cauca por brindarme un entorno de aprendizaje donde he crecido no solo en conocimientos, sino también en valores.

Robinson Albeiro Tijo Gonzalez

Quiero expresar mi más profundo agradecimiento a la profesora Mary Luz Ojeda Solarte y al profesor José Gabriel Pérez Canencio por su dedicación, orientación y valiosas enseñanzas a lo largo de mi investigación. También quiero extender mi gratitud a la Universidad Unidad Central del Valle del Cauca por brindarme la oportunidad de aprender y crecer en este entorno académico.

Un agradecimiento especial a mi compañero de tesis, Robinson Albeiro Tijo Gonzalez, por su colaboración incansable, intercambio de ideas y trabajo en equipo, que enriquecieron enormemente este proyecto.

Jaen Fellini Echavarria Tijo

GLOSARIO

PWA: (**Progressive Web Apps).** Es una aplicación web que ofrece características de aplicaciones móviles, como instalación en el dispositivo, funcionamiento offline y notificaciones, mientras se accede a través de un navegador web.

Angular: Es un framework de desarrollo de aplicaciones web en JavaScript/TypeScript que facilita la creación de aplicaciones interactivas y dinámicas, mediante el uso de componentes y una arquitectura estructurada.

CSS: Es un lenguaje de diseño utilizado en desarrollo web para controlar la apariencia y presentación visual de páginas y elementos HTML.

HTML: Es el lenguaje fundamental de marcado utilizado para crear y estructurar el contenido de las páginas web.

TypeScript: Es un lenguaje de programación que amplía JavaScript al agregar tipos estáticos y características avanzadas de desarrollo, lo que mejora la seguridad y mantenibilidad del código.

Axios: Es una librería de JavaScript utilizada para realizar solicitudes HTTP desde el navegador o Node.js. Ofrece una interfaz sencilla y conveniente para trabajar con APIs y manejar respuestas de manera eficiente.

Docker: Es una plataforma de virtualización ligera que permite empacar, distribuir y ejecutar aplicaciones en entornos aislados, llamados contenedores, garantizando la consistencia y portabilidad en diferentes sistemas.

Docker Compose: Es una herramienta que permite definir y administrar aplicaciones compuestas por varios contenedores de Docker de manera sencilla, utilizando un archivo de configuración único.

Service Workers: Son scripts de JavaScript que se ejecutan en segundo plano para habilitar características como notificaciones push, actualizaciones en segundo plano y el almacenamiento en caché, lo que permite que la PWA funcione de manera eficiente incluso sin conexión a Internet.

VPS: Es un servidor virtual que se ejecuta en un servidor físico compartido con otros servidores virtuales. Los VPS se utilizan para alojar sitios web, aplicaciones, bases de datos y otros servicios.

RESUMEN

En atención a los avances de las tecnologías del momento, se ha tomado la decisión de implementar una estrategia de migración de la aplicación móvil que solo funciona en dispositivos con sistema operativo Android a un sistema multiplataforma que represente beneficios significativos para los participantes al Taller de Sensibilización Frente al Cambio Climático, como, por ejemplo:

Un mayor alcance en los participantes al Taller: Se podrá acceder a un mayor número de participantes potenciales que utilizan diferentes dispositivos y sistemas operativos, aumentando la visibilidad y la exposición de la aplicación, lo que puede llevar a un mayor número de descargas e interacciones con los usuarios.

Mejora en la experiencia del usuario: Al permitir que los usuarios utilicen la aplicación en cualquier dispositivo, independientemente de su sistema operativo, se mejora la experiencia del usuario y se aumenta la satisfacción del participante al Taller. Los usuarios apreciarán la flexibilidad y la accesibilidad de poder acceder a la aplicación desde cualquier dispositivo.

Ahorro de tiempo y recursos: Al migrar una aplicación móvil a multiplataforma, se ahorra tiempo y recursos al no tener que desarrollar y mantener versiones separadas de la aplicación para cada sistema operativo. Esto simplifica el proceso de desarrollo y reduce los costos asociados con la creación y mantenimiento de múltiples versiones de la aplicación.

Facilidad de actualización y mantenimiento: Al tener una única versión de la aplicación que funciona en múltiples plataformas, se facilita la tarea de actualizar y mantener la aplicación. Los desarrolladores pueden implementar cambios y mejoras de forma más eficiente y rápida, lo que garantiza que los usuarios siempre tengan acceso a la versión más reciente de la aplicación.

ABSTRACT

In response to current technological advances, the decision has been made to implement a migration strategy from the mobile application that only works on devices with Android operating system to a multiplatform system that represents significant benefits for the participants of the Climate Change Awareness Workshop, such as, for example:

Increased reach of workshop participants: A larger number of potential participants using different devices and operating systems will be accessible, increasing the visibility and exposure of the application, which can lead to a higher number of downloads and interactions with users

Improved user experience: Enabling users to use the application on any device, regardless of their operating system, improves the user experience and increases participant satisfaction with the Workshop. Users will appreciate the flexibility and accessibility of being able to access the application from any device.

Time and resource savings: By migrating a mobile application to cross-platform, you save time and resources by not having to develop and maintain separate versions of the application for each operating system. This simplifies the development process and reduces the costs associated with creating and maintaining multiple versions of the application.

Ease of upgrade and maintenance: By having a single version of the application that works on multiple platforms, the task of upgrading and maintaining the application is made easier. Developers can implement changes and enhancements more efficiently and quickly, ensuring that users always have access to the latest version of the application.

INTRODUCIÓN

En la actualidad, el desarrollo de aplicaciones móviles se ha convertido en una tendencia cada vez más importante debido a la creciente dependencia de los dispositivos móviles en la vida diaria de las personas y especialmente en las actividades que se ejecutan cuando se desarrolla un Taller de Sensibilización Frente al Cambio Climático. Sin embargo, la diversidad de plataformas móviles existentes, como iOS y Android, representa un desafío para los desarrolladores a la hora de crear aplicaciones que sean compatibles con todos los sistemas operativos.

En este contexto, el presente trabajo de grado se enfoca en la migración de una aplicación móvil desarrollada inicialmente para la plataforma Android a un sistema multiplataforma que permita su ejecución en diferentes dispositivos y sistemas operativos. Se abordarán las ventajas y desventajas de la migración, así como las herramientas y técnicas necesarias para lograr una transición efectiva y exitosa.

El objetivo principal de este proyecto es facilitar la accesibilidad de la aplicación a un mayor número de usuarios del Taller de Sensibilización Frente al Cambio Climático, ampliando su alcance y brindando una experiencia de usuario consistente en todas las plataformas. A través de este estudio, se pretende mostrar la importancia de la adaptabilidad y versatilidad en el desarrollo de aplicaciones móviles para satisfacer las necesidades cambiantes de los usuarios en un entorno tecnológico en constante evolución.

TABLA DE CONTENIDO

CAPITULO 1. GENERALIDADES 14
1.1 EL PROBLEMA 14
1.1.1 DESCRIPCIÓN 14
1.1.2 FORMULACIÓN 15
1.2 JUSTIFICACIÓN 16
1.3 OBJETIVOS 17
1.3.1 GENERAL 17
1.3.2 ESPECÍFICOS 17
CAPITULO 2. MARCOS DE REFERENCIA 18
2.1 MARCO TEÓRICO 18
2.1.1 DESPLIEGUE DE SOFTWARE 18
2.1.2. ENTORNOS DE TRABAJO 21
2.1.3 MARCO DE TRABAJO SCRUM 25
2.1.3.1. COMPONENTES DE SCRUM. 27
CAPITULO 3. DISEÑO METODOLOGICO DEL PROYECTO 36
3.1. TIPO Y ENFOQUE DE LA INVESTIGACIÓN 36
3.2. METODOLOGÍA 36
3.2.1 FASES DEL PROCESO METODOLÓGICO APLICADO 36
3.2.2 INSTRUMENTOS PARA LA RECOLECCIÓN DE LA INFORMACIÓN. 37
CAPITULO 4. INGENIERIA DEL PROYECTO 38
4.1. SITUACIÓN ACTUAL DE LA VERSIÓN ORIGINAL DEL APLICATIVO: 38
4.2. PROCESO DE DESARROLLO BASADO EN SCRUM 39
4.2.1 CREACIÓN DEL EQUIPO SCRUM 39
4.2.2 PLANIFICACIÓN Y DEFINICIÓN DEL PROYECTO 41
4.2.2.1 SPRINT 0 45
4.3 ARQUITECTURA DEL SISTEMA 48
4.4 DESARROLLO DE LA APLICACIÓN MÓVIL 55
4.5 MODELO DE LA BASE DE DATOS 65
4.5.1 DEFINICIÓN DE MODELO ENTIDAD RELACIÓN 66
4.6. PROTOTIPADO. DISEÑO DE LAS INTERFACES GRÁFICAS DE USUARIO FINAL 67
4.6.1. DESCRIPCIÓN DE DISEÑO. 68

4.4.2 DESCRIPCIÓN GENERAL DE LAS INTERFACES GRÁFICAS DE USUARIO. GUI´S 77
CAPITULO 5. CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO 78
5.1 CONCLUSIONES 78
5.2 RECOMENDACIONES 79
5.3. TRABAJO FUTURO 79
REFERENCIAS CONSULTADAS 80

INDICE DE ILUSTRACIONES

Ilustración 1 Fases de Devops 19
Ilustración 2 Marco de trabajo Scrum 26
Ilustración 3 Proceso metodológico 36
Ilustración 4 Comparativa entre iOS y Android 39
Ilustración 5. Tablero scrum describiendo el product backlog 45
Ilustración 6. Vista general del sprint “0” 47
Ilustración 7. Diagrama de componentes del sistema 53
Ilustración 8. Vista global de los componentes del proyecto 57
Ilustración 9. Vista ampliada de la carpeta “public” 60
Ilustración 10. Vista ampliada de la pantalla de inicio o login 63
Ilustración 11. Vista ampliada de los recursos utilizados en la aplicación 65
Ilustración 12. Modelo de la base de datos 66
Ilustración 13. Interfaces gráficas de usuario 68
Ilustración 14. Interfaz de acceso 69
Ilustración 15. Interfaz gráfica notificación de acceso. 71
Ilustración 16. Ilustración 3. Interfaz gráfica de inicio. 72
Ilustración 17. . Interfaz gráfica calcular huella 73
Ilustración 18. Interfaz gráfica cuestionario. 74
Ilustración 19. Interfaz gráfica ranking 75
Ilustración 20. Interfaz gráfica de chat 76

CAPITULO 1. GENERALIDADES

1.1 EL PROBLEMA

1.1.1 DESCRIPCIÓN

Dentro del Grupo de Investigación GIGAE3D de la Facultad de Ingeniería, los profesores José Gabriel Pérez Canencio y Mary Luz Ojeda Solarte han desarrollado un recurso educativo digital (taller Interactivo), con el propósito de facilitar la socialización y formación de la población en el ámbito de la conciencia ambiental y los problemas asociados. Esta herramienta tiene como objetivo permitir esfuerzos individuales y colectivos para abordar problemas directamente relacionados con la administración eficiente del consumo de energía.

El taller interactivo digital se basa en un marco de trabajo colaborativo centrado en la educación ambiental y la eficiencia energética en organizaciones. Logra esto a través de talleres que se centran en el cambio climático, poniendo énfasis en los hábitos de consumo de los usuarios de Tecnologías de la Información y la Comunicación (TIC).

Cada taller consta de dos conferencias sobre el tema mencionado y dos sesiones prácticas que incluyen aplicaciones web y móviles desarrolladas por los profesores mencionados en colaboración con estudiantes. Estas sesiones abarcan el cálculo de la huella de carbono del alcance 2 y un juego entretenido para evaluar el conocimiento adquirido sobre los temas tratados. Este taller, titulado "Taller de Sensibilización frente al Cambio Climático", sigue las directrices establecidas en la Agenda de Acción Climática Global de las Naciones Unidas.

En la actualidad, el taller se ha implementado en diversas universidades del país, ubicadas en diferentes ciudades. Además, se ha participado en diversos congresos de alcance nacional e internacional, lo que ha brindado oportunidades para presentar el taller a audiencias de otros países.

El primer paso para la implementación de un taller implica crearlo como un evento en una aplicación web, desde donde se gestiona la información de los participantes y las actividades de cada evento.

La aplicación móvil se encarga de llevar a cabo las sesiones prácticas mencionadas anteriormente. Una de estas sesiones se refiere al cálculo de la huella de carbono del alcance 2, que se realiza en dos oportunidades. Otro módulo consiste en un juego lúdico de preguntas y respuestas, cuyos resultados se presentan en un módulo de clasificación, mientras que un chatbot responde preguntas y se entrena de manera constante.

En la actualidad, la aplicación está desarrollada exclusivamente para dispositivos Android, lo que limita el acceso al taller a un público más reducido.

Con el transcurso del tiempo, las aplicaciones han experimentado ciertos problemas que requieren correcciones para garantizar la confiabilidad de la información sobre el consumo eléctrico enviada por los participantes al servidor.

1.1.2 FORMULACIÓN

¿Cómo realizar con éxito la migración y actualización del aplicativo web y móvil del taller de sensibilización frente al cambio climático, logrando su adaptación efectiva a un entorno multiplataforma?

1.2 JUSTIFICACIÓN

La migración de esta aplicación tiene una importancia significativa para los organizadores de los talleres de sensibilización, ya que les brinda la oportunidad de llegar a una audiencia más amplia en sus eventos.

Esta migración no solo posibilita que los participantes accedan a los talleres desde dispositivos Android, sino también desde dispositivos iOS, sin ninguna restricción en relación al tamaño de pantalla de los dispositivos. Así, se logra una mayor flexibilidad y accesibilidad, enriqueciendo la experiencia de los usuarios y maximizando el alcance y efectividad de los talleres de sensibilización frente al cambio climático.

Adicionalmente, es una oportunidad para ampliar la cobertura del taller dado que se posibilita que la mayoría de los participantes realicen las prácticas de la actividad en cualquier tipo de dispositivo, bien sea, una Tablet, un pc de escritorio, un portátil o un dispositivo móvil con sistema operativos Android o iOS.

En síntesis, migrar la aplicación móvil de Android a multiplataforma es una decisión estratégica que tendrá un impacto positivo significativo en la visibilidad, la accesibilidad y la satisfacción de los participantes al Taller. Además, simplifica el proceso de desarrollo y mantenimiento, lo que puede resultar en ahorros de tiempo y recursos a largo plazo, siendo sostenible.

1.3 OBJETIVOS

1.3.1 GENERAL

Migrar y actualizar la aplicación web y móvil del taller de sensibilización frente al cambio climático para adaptarla a un entorno multiplataforma.

1.3.2 ESPECÍFICOS

- Realizar un análisis exhaustivo de las funcionalidades actuales de las aplicaciones para determinar las áreas que requieren actualización.

- Desarrollar y agregar nuevas funcionalidades a las aplicaciones web y móvil.

- Adaptar las aplicaciones a un entorno multiplataforma que permita su ejecución en dispositivos con sistemas operativos Android y iOS.

- Realizar pruebas exhaustivas y entregar las aplicaciones, asegurando su pleno funcionamiento en los talleres de sensibilización frente al cambio climático.

CAPITULO 2. MARCOS DE REFERENCIA

2.1 MARCO TEÓRICO

2.1.1 DESPLIEGUE DE SOFTWARE

Definición. Se define como el proceso mediante el cual una aplicación de software es introducida (versión inicial) o actualizada en un entorno específico, garantizando su correcto funcionamiento y disposición para los usuarios finales. Este proceso requiere una cuidadosa planificación e implementación, teniendo en cuenta una secuencia de pasos, los cuales pueden experimentar variaciones tanto en su naturaleza como en su orden, según las características específicas del entorno en el que se lleva a cabo y, especialmente, de las peculiaridades inherentes a cada aplicación en particular.

A lo largo del ciclo de vida de una aplicación, el despliegue de software involucra diversas tareas, que van desde la configuración de servidores hasta la carga de la aplicación, la realización de pruebas, la migración de datos y la actualización de versiones anteriores. Algunos procesos, como la configuración de los servidores, se llevan a cabo únicamente en la tapa inicial o primer despliegue, mientras que otros se repiten en cada uno de los despliegues.

En el pasado, los despliegues de software se gestionaban “a mano”. No obstante, dada la creciente complejidad y constante evolución de los sistemas de software en la actualidad, es necesario pensar en un nivel de automatización cada vez mayor. Esto se debe a la velocidad con la que evolucionan los sistemas de software y los desafíos que esto presenta para los procesos de despliegue (Arcangeli et al., 2015).

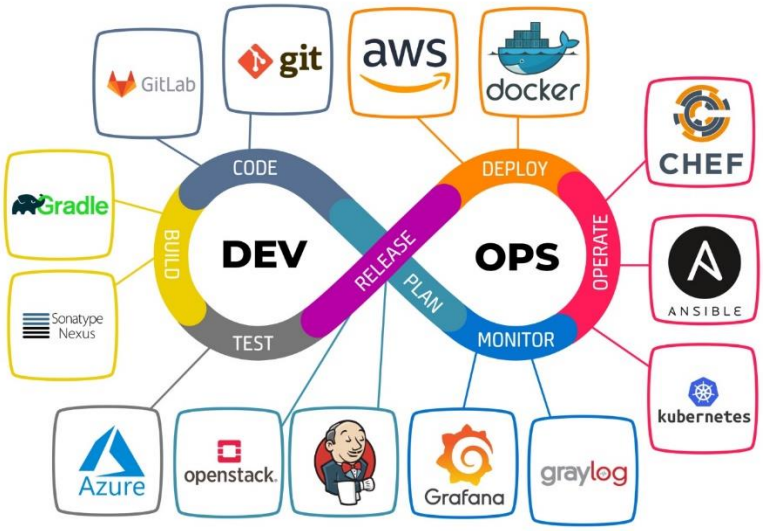
Según (Zambrano et al., 2021), automatizar el proceso de despliegue es sumamente crucial si lo que se busca es la optimización de los procesos empresariales (p. 2). Un caso exitoso presentado por (Muñoz et al., 2020), muestra cómo una pequeña empresa de desarrollo logró incrementar de manera significativa el número de despliegues a producción mediante la automatización, pasando así de cuatro a veinte

despliegues mensuales y logrando al mismo tiempo disminuir el número de despliegues fallidos (p. 9).

En empresas dedicadas al desarrollo de software, es frecuente contar con personal especializado en la gestión de despliegues. Esta práctica tiene como objetivo liberar al equipo de desarrollo de ciertas responsabilidades y mejorar los tiempos de despliegue. Este equipo se encarga de brindar apoyo mediante la automatización de algunos procesos de despliegue, con el fin de mejorar la calidad del software, facilitar la entrega continua y reducir los ciclos de desarrollo.

La cultura DevOps propone fortalecer la colaboración entre los equipos de desarrollo y operaciones, comúnmente conocido como el equipo de despliegues, a lo largo de todo el ciclo de vida del desarrollo. En la Figura 1, se observan las distintas fases que conforman DevOps y las herramientas asociadas a cada una. Notablemente, se destaca la presencia de un símbolo de infinito, simbolizando que este proceso es continuo a lo largo de toda la vida de la aplicación.

Ilustración 1 Fases de Devops



Fuente: (Unity, 2020)

La Integración Continua, conocida como CI por sus siglas en inglés, se describe según (Alcásiga et al., 2022) como una práctica de desarrollo de software que tiene como objetivo fusionar el código proveniente de diferentes desarrolladores tan frecuentemente como sea posible. Esto se realiza con el propósito de identificar errores de integración de manera temprana. Cabe destacar que cada integración se somete a una validación mediante una compilación automatizada, la cual incorpora el proceso de pruebas (p. 32).

En el marco de una metodología ágil, caracterizada por la generación de entregables incrementales con diversas funcionalidades construidas por diferentes desarrolladores en cada sprint, se vuelve esencial la implementación de la Integración Continua. Esta práctica, estrechamente alineada con el Despliegue Continuo, posibilita la incorporación, corrección de errores y configuración del código en un entorno de producción (Alcásiga et al., 2022, p. 33). Estos procedimientos constituyen pilares esenciales de la fase de lanzamiento en DevOps, siendo común la utilización de la herramienta Jenkins para llevarlos a cabo.

Es importante destacar que el despliegue de software puede automatizarse tanto como se requiera. Algunas organizaciones prefieren realizar los despliegues de software de forma manual, mientras que otras optan por la automatización completa de este proceso. Debe entenderse que no existe una única manera de abordar el despliegue de software; más bien, es algo que varía según las necesidades y preferencias específicas de cada empresa. Lo crucial es encontrar el enfoque que mejor se adapte a cada caso, ya sea manual, automatizado o una combinación de ambos, con el objetivo de mejorar la eficiencia y la fiabilidad en cada implementación.

Independientemente de la estrategia de despliegue que se elija, es fundamental contar con una documentación detallada y accesible dentro de la empresa. Esta práctica no solo facilita la integración de nuevos miembros al equipo, sino que también contribuye a la coherencia y consistencia en la ejecución del proceso. Al disponer de una guía clara, los miembros del equipo pueden acoplarse rápidamente a las

directrices establecidas, optimizando así la eficiencia y minimizando posibles errores en el proceso de despliegue.

2.1.2. ENTORNOS DE TRABAJO

En el desarrollo de cualquier producto de software nuevo, desde su codificación hasta su disposición final, el software atraviesa diversos entornos o ambientes que resultan fundamentales en cada fase del proyecto.

El entorno de desarrollo, siendo el más inestable, es donde todo comienza. Los diferentes desarrolladores inician codificando los requerimientos funcionales y no funcionales, para posteriormente probarlos en sus propias máquinas o en servidores locales. Por esta razón, el entorno de desarrollo también se conoce comúnmente como ambiente local. En esta fase del proyecto, los despliegues de software propiamente dichos no suelen implementase, ya que cada desarrollador ejecuta el proyecto de manera local o lo envía a un servidor local para su ejecución. En la mayoría de los casos, este proceso se lleva a cabo de manera informal y rara vez está automatizado o documentado.

El entorno de desarrollo se compone de una amplia gama de herramientas que pueden variar según la naturaleza del proyecto o las preferencias del equipo hacia herramientas privativas u opensource. Tampoco debemos pasar por alto la influencia del sistema operativo, ya que en ocasiones puede restringir la elección de herramientas disponibles. En los equipos de desarrollo, es común que los desarrolladores tengan preferencias personales basadas en su gusto o experiencia con alguna herramienta en específico. Este fenómeno también se puede observar en la elección de sistemas operativos, dado que la diversidad existente permite a los desarrolladores optar por aquel con el que se sientan más cómodos.

Dentro del entorno de desarrollo, se encuentran herramientas fundamentales como editores de código o Entornos de Desarrollo Integrados (IDEs), Software Development Kits (SDKs) para diversos lenguajes de programación, navegadores y herramientas destinadas a realizar pruebas de API. Además, el entorno puede incluir motores de bases de datos y emuladores diseñados para dispositivos móviles, tablets, televisores o smartwatches.

En el mercado podemos encontrar una amplia variedad de herramientas de desarrollo, que van desde soluciones gratuitas hasta software de pago. Asimismo, existen licencias comunitarias que ofrecen funcionalidades básicas, aunque con restricciones, diseñadas especialmente para equipos más reducidos.

Los editores de código o en su defecto los IDEs son las herramientas con las cuales los desarrolladores pasan la mayor parte de su tiempo. Por ende, es crucial que estas herramientas sean cómodas y eficientes, proporcionando a los desarrolladores un entorno propicio para trabajar de manera productiva durante extensas jornadas.

Actualmente existen Entornos de Desarrollo Integrados (IDEs) para prácticamente todos los lenguajes de programación, incluso existen varias opciones para un mismo lenguaje en particular. Entre los IDEs ampliamente reconocidos se encuentran Visual Studio, Eclipse e IntelliJ IDEA, destacándose por su popularidad y funcionalidades. Asimismo, existen IDEs especializados, pensados para abordar las necesidades específicas de ciertos lenguajes o para llevar a cabo tareas particulares, como es el caso de Android Studio, enfocado en el desarrollo de aplicaciones móviles, o WebStorm, diseñado para el desarrollo web.

Aunque tanto los editores de código como los IDEs comparten el objetivo fundamental de facilitar la escritura de código, los IDEs se destacan por ser más completos, proporcionando una serie de herramientas adicionales que simplifican el proceso de desarrollo de software. No obstante, algunos editores de código tienen la

capacidad de ampliar sus funcionalidades mediante plugins, llegando incluso a equipararse en ciertos aspectos con las capacidades de un IDE.

De acuerdo con las encuestas anuales realizadas por Stack Overflow, Visual Studio Code se ha posicionado como el editor de código más utilizado a nivel mundial, no solo ha sido el más utilizado en el año 2023, con un porcentaje de uso del 73,71, sino que también ha mantenido dicha posición a lo largo de varios años. La clave de su grandeza reside, sin lugar a duda, en dos pilares fundamentales:

- La sólida comunidad comprometida que tiene detrás donde miles de desarrolladores contribuyen activamente para crear extensiones, temas y otras herramientas que amplían las funcionalidades del editor.
- El respaldo por parte de Microsoft que ha permitido mejorar continuamente el rendimiento, estabilidad y seguridad del editor.

Microsoft, una empresa reconocida por sus destacados servicios en la nube a través de Azure y su amplio ecosistema, ha llevado a cabo un giro innovador al presentar CodeSpaces, una herramienta diseñada para optimizar la colaboración en equipos de desarrollo. Lo excepcional de esta solución radica en su capacidad para facilitar la colaboración en tiempo real, sobrepasando los límites convencionales al ofrecer un entorno de desarrollo completamente remoto alojado en la nube. Con solo acceso a Internet, los desarrolladores pueden comenzar a trabajar sin preocuparse por instalaciones o recursos de hardware, ya que CodeSpaces se encarga de proporcionar todo lo necesario. Además, la flexibilidad que otorga a los desarrolladores al poder acceder al entorno de trabajo desde cualquier lugar del mundo, junto con la capacidad de crear diversos entornos de trabajo instantáneamente, agrega un nivel superior de versatilidad a esta herramienta.

Según se menciona en (“GitHub Brings Out GitHub Codespaces, GitHub Discussions, GitHub Private Instances And More,” 2020), GitHub ha decidido integrar el editor de Visual Studio Code en su versión basada en el navegador de Codespaces.

Esto permite a los desarrolladores navegar por el código de manera libre, utilizar extensiones y acceder a la terminal. Además, si un desarrollador prefiere utilizar su propio editor de código en su versión de escritorio, solo necesita iniciar un espacio de código en GitHub y conectarse a él desde su estación de trabajo.

Codespaces, en resumen, se basa en los Development Containers o contenedores de desarrollo, que son esencialmente contenedores de Docker diseñados específicamente para el desarrollo de proyectos. Este enfoque proporciona un entorno de desarrollo consistente y reproducible, facilitando la colaboración y eliminando las barreras relacionadas con la configuración del entorno de desarrollo. En definitiva, Visual Studio Code y CodeSpaces ofrecen una solución integral y avanzada para equipos de desarrollo que buscan eficiencia, flexibilidad y colaboración sin restricciones geográficas.

Una práctica reciente que ha demostrado ser efectiva consiste en establecer una gran similitud entre el entorno de desarrollo y el entorno de producción. En un tiempo atrás, era bastante común encontrar errores al momento de desplegar el software en el entorno final debido a las diferencias que existían entre los ambientes de desarrollo y producción; de allí la famosa frase “es que en mi máquina funciona”. Las máquinas virtuales, por su parte, surgieron como solución para mitigar este inconveniente, proporcionando a los desarrolladores el mismo conjunto de herramientas y versiones empaquetadas en una imagen fácil de importar, según lo menciona (Aznar et al., 2023, p. 8). Esto no solo reducía el tiempo de acondicionamiento para nuevos miembros del equipo, sino que también minimizaba los inconvenientes al momento de desplegar en el entorno real.

Sin embargo, a pesar de ser una opción viable y ampliamente utilizada, es importante destacar que, como lo menciona (Aznar et al., 2023), las máquinas virtuales requerían de una gran cantidad de recursos de hardware (p. 8). Como resultado, esta práctica ha ido siendo reemplazada progresivamente por el uso de herramientas más optimizadas y avanzadas.

Es importante señalar que en algunas empresas de desarrollo se acostumbra a contar con un rol específico o un equipo de Control de Calidad (QA), cuya responsabilidad radica en poner a prueba las diversas funcionalidades en el entorno de pruebas. Aunque no se trata de una fase estrictamente indispensable, su presencia resulta fundamental cuando el objetivo es lanzar un producto de software de alta calidad. Personalmente, consideramos que incorporar un equipo de QA constituye una buena práctica que aporta fiabilidad al producto final.

Además de los entornos de desarrollo y pruebas, existen otros entornos importantes como el de preproducción y producción. Aunque el entorno de preproducción y las pruebas no son obligatorios, su inclusión mejora significativamente la calidad del software a entregar. Muchas empresas optan por implementar estos entornos en su proceso de desarrollo de software para garantizar un producto final de alta calidad.

2.1.3 MARCO DE TRABAJO SCRUM

Con el fin de contextualizar a cerca del método para el desarrollo de software que se empleó en este trabajo, se hace referencia a la fuente principal del método llamado scrum y que se ha recuperado del portal principal presentado en la web (Francia Huambachano, 2017) que textualmente dice:

Scrum es un proceso de gestión que reduce la complejidad en el desarrollo de productos para satisfacer las necesidades de los clientes. La gerencia y los equipos de Scrum trabajan juntos alrededor de requisitos y tecnologías para entregar productos funcionando de manera incremental usando el empirismo.

Scrum es un marco de trabajo de procesos que ha sido usado para gestionar el desarrollo de productos complejos desde principios de los años 90. Scrum no es un proceso o una técnica para construir productos; en lugar de eso, es un marco de trabajo

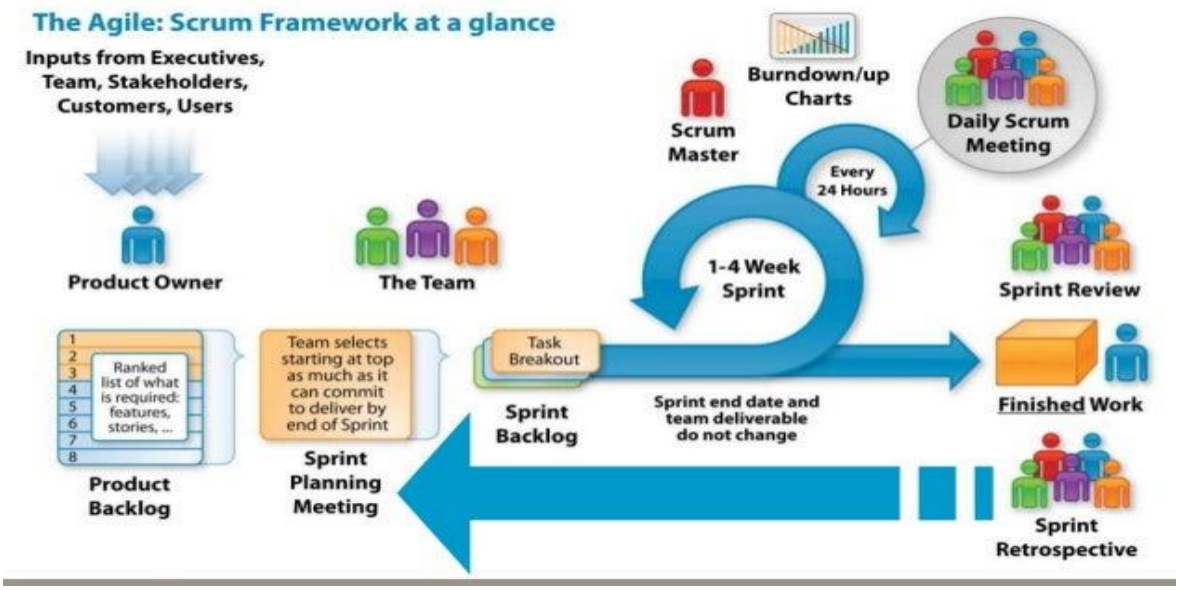
dentro del cual se pueden emplear varios procesos y técnicas. Scrum muestra la eficacia relativa de las prácticas de gestión de producto y las prácticas de desarrollo de modo que podamos mejorar.

El marco de trabajo Scrum consiste en los Equipos Scrum y sus roles, eventos, artefactos y reglas asociadas. Cada componente dentro del marco de trabajo sirve a un propósito específico y es esencial para el éxito de Scrum. De acuerdo (Shcwaber & Sutherland, 2020), scrum emplea un enfoque iterativo e incremental para optimizar la predictibilidad y el control del riesgo.

Scrum es simple, no es una gran colección de partes y componentes obligatorios definidos de manera prescriptiva. Scrum no es una metodología, Scrum está basado en un modelo de proceso empírico con respeto a las personas y basado en la auto organización de los equipos para lidiar con lo imprevisible y resolver problemas complejos inspeccionando y adaptando continuamente.

El siguiente gráfico representa Scrum como lo describen Ken Schwaber y Jeff Sutherland

Ilustración 2 Marco de trabajo Scrum



Fuente: www.atlassian.com/jira-software/scrum

2.1.3.1. COMPONENTES DE SCRUM.

Scrum Team. La unidad fundamental de Scrum es un pequeño equipo de personas, un Scrum Team. El Scrum team consta de un scrum master, un product owner y developers.

Dentro de un Scrum Team, no hay sub equipos ni jerarquías. Es una unidad cohesionada de profesionales enfocados en un objetivo a la vez: el Objetivo del Producto.

Los Scrum Teams son multifuncionales, lo que significa que los miembros tienen todas las habilidades necesarias para crear valor en cada Sprint. También se auto gestionan, lo que significa que deciden internamente quién hace qué, cuándo y cómo.

El Scrum Team es lo suficientemente pequeño como para seguir siendo ágil y lo suficientemente grande como para completar un trabajo significativo dentro de un Sprint, generalmente 10 personas o menos. En general, los equipos más pequeños se comunican mejor y son más productivos. Si los Scrum Teams se vuelven demasiado grandes, deberían considerar reorganizarse en múltiples Scrum Teams cohesivos, cada uno enfocado en el mismo producto. Por lo tanto, deben compartir el mismo objetivo del producto, el product backlog y el product owner.

El Scrum Team es responsable de todas las actividades relacionadas con el producto, desde la colaboración de los interesados, la verificación, el mantenimiento, la operación, la experimentación, la investigación y el desarrollo, y cualquier otra cosa que pueda ser necesaria.

Están estructurados y empoderados por la organización para gestionar su propio trabajo.

Trabajar en Sprints a un ritmo sostenible mejora el enfoque y la consistencia del Scrum Team.

Todo el Scrum Team es responsable de crear un incremento valioso y útil en cada Sprint. Scrum define tres responsabilidades específicas dentro del Scrum Team: los Developers, el Product Owner y el Scrum Master.

Developers. Las personas del Scrum team que se comprometen a crear cualquier aspecto de un Incremento utilizable en cada Sprint son Developers. Las habilidades específicas que necesitan los Developers suelen ser amplias y variarán según el ámbito de trabajo.

Sin embargo, los Developers siempre son responsables de:

- Crear un plan para el Sprint, el Sprint Backlog
- Inculcar calidad al adherirse a una definición de terminado
- Adaptar su plan cada día hacia el objetivo del Sprint; y
- Responsabilizarse mutuamente como profesionales.

Product Owner. El Product Owner es responsable de maximizar el valor del producto resultante del trabajo del Scrum Team. La forma en que esto se hace puede variar ampliamente entre organizaciones, Scrum Teams e individuos.

El Product Owner también es responsable de la gestión efectiva del Product Backlog, lo que incluye:

- Desarrollar y comunicar explícitamente el Objetivo del Producto
- Crear y comunicar claramente los elementos del Product Backlog
- Ordenar los elementos del Product Backlog; y,
- Asegurarse de que el Product Backlog sea transparente, visible y se entienda.

El Product Owner puede realizar el trabajo anterior o puede delegar la responsabilidad en otros. Independientemente de ello, el Product Owner sigue siendo el responsable de que el trabajo se realice.

Para que los Product Owners tengan éxito, toda la organización debe respetar sus decisiones. Estas decisiones son visibles en el contenido y el orden del Product Backlog, y a través del Incremento inspeccionable en la Sprint Review.

El Product Owner es una persona, no un comité. El Product Owner puede representar las necesidades de muchos interesados en el Product Backlog. Aquellos que quieran cambiar el Product Backlog pueden hacerlo intentando convencer al Product Owner.

Scrum Master. El Scrum Master ayuda a todos a comprender la teoría y la práctica de Scrum, tanto dentro del Scrum Team como de la organización.

El Scrum Master es responsable de lograr la efectividad del Scrum Team. Lo hace apoyando al Scrum Team en la mejora de sus prácticas, dentro del marco de trabajo de Scrum.

Los Scrum Masters son verdaderos líderes que sirven al Scrum Team y a la organización en general.

El Scrum Master sirve al Scrum Team de varias maneras, que incluyen:

● Guiar a los miembros del equipo en ser auto gestionados y multifuncionales.

● Ayudar al Scrum Team a enfocarse en crear Incrementos de alto valor que cumplan con la Definición de Terminado.

● Procurar la eliminación de impedimentos para el progreso del Scrum Team; y,

● Asegurarse de que todos los eventos de Scrum se lleven a cabo y sean positivos, productivos y se mantengan dentro de los límites de tiempo recomendados.

El Scrum Master sirve al Product Owner de varias maneras, que incluyen:

● Ayudar a encontrar técnicas para una definición efectiva de Objetivos del Producto y la gestión del Product Backlog.

● Ayudar al Scrum Team a comprender la necesidad de tener elementos del Product Backlog claros y concisos.

● Ayudar a establecer una planificación empírica de productos para un entorno complejo; y,

● Facilitar la colaboración de los interesados según se solicite o necesite.

El Scrum Master sirve a la organización de varias maneras, que incluyen:

● Liderar, capacitar y guiar a la organización en su adopción de Scrum

● Planificar y asesorar implementaciones de Scrum dentro de la organización

● Ayudar a los empleados y los interesados a comprender y aplicar un enfoque empírico para el trabajo complejo; y,

● Eliminar las barreras entre los interesados y los Scrum Team.

Eventos de Scrum. El Sprint es un contenedor para todos los demás eventos.

Cada evento en Scrum es una oportunidad formal para inspeccionar y adaptar los artefactos Scrum. Estos eventos están diseñados específicamente para habilitar la transparencia requerida.

No operar cualquier evento según lo prescrito resulta en la pérdida de oportunidades para inspeccionar y adaptarse. Los eventos se utilizan en Scrum para crear regularidad y minimizar la necesidad de reuniones no definidas en Scrum. Lo óptimo es que todos los eventos se celebren al mismo tiempo y en el mismo lugar para reducir la complejidad.

El Sprint. Los Sprints son el corazón de Scrum, donde las ideas se convierten en valor. Son eventos de duración fija de un mes o menos para crear consistencia. Un nuevo Sprint comienza inmediatamente después de la conclusión del Sprint anterior.

Todo el trabajo necesario para lograr el Objetivo del Producto, incluido la Sprint Planning, Daily Scrums, Sprint Review y Sprint Retrospective, ocurre dentro de los Sprints.

Durante el Sprint:

● No se realizan cambios que pongan en peligro el Objetivo del Sprint.

● La calidad no disminuye.

● El Product Backlog se refina según sea necesario; y,

● El alcance se puede aclarar y renegociar con el Product Owner a medida que se aprende más.

Los Sprints permiten la previsibilidad al garantizar la inspección y adaptación del progreso hacia un Objetivo del Producto al menos cada mes calendario. Cuando el horizonte de un Sprint es demasiado largo, el Objetivo del Sprint puede volverse inválido, la complejidad puede crecer y el riesgo puede aumentar.

Se pueden emplear Sprints más cortos para generar más ciclos de aprendizaje y limitar el riesgo de costo y esfuerzo a un período de tiempo menor. Cada Sprint puede considerarse un proyecto corto.

Existen varias prácticas para pronosticar el progreso, como el trabajo pendiente (burn-downs), trabajo completado (burn-ups) o flujos acumulativos (cumulative flows). Si bien han demostrado su utilidad, no reemplazan la importancia del empirismo. En entornos complejos, se desconoce lo que sucederá.

Solo lo que ya ha sucedido se puede utilizar para la toma de decisiones con miras al futuro.

Un Sprint podría cancelarse si el Objetivo del Sprint se vuelve obsoleto. Solo el Product Owner tiene la autoridad para cancelar el Sprint.

Sprint Planning. La Sprint Planning inicia el Sprint al establecer el trabajo que se realizará para el Sprint. El Scrum Team crea este plan resultante mediante trabajo colaborativo.

El Product Owner se asegura de que los asistentes estén preparados para discutir los elementos más importantes del Product Backlog y cómo se relacionan con el Objetivo del Producto. El Scrum Team también puede invitar a otras personas a asistir a la Sprint Planning para brindar asesoramiento.

La Sprint Planning aborda los siguientes temas:

Tema uno: ¿Por qué es valioso este Sprint? El Product Owner propone cómo el producto podría Incrementar su valor y utilidad en el Sprint actual. Luego, todo el Scrum Team colabora para definir un Objetivo del Sprint que comunica por qué el Sprint es valioso para los interesados. El Objetivo del Sprint debe completarse antes de que termine la Sprint Planning

Tema dos: ¿Qué se puede hacer en este Sprint? A través de una conversación con el Product Owner, los Developers seleccionan elementos del Product Backlog para incluirlos en el Sprint actual. El Scrum Team puede refinar estos elementos durante este proceso, lo que aumenta la comprensión y la confianza. Seleccionar cuánto se puede completar dentro de un Sprint puede ser un desafío. Sin embargo, cuanto más sepan los Developers sobre su desempeño pasado, su capacidad actual y su Definición de Terminado, más confiados estarán en sus pronósticos para el Sprint.

Tema tres: ¿Cómo se realizará el trabajo elegido? Para cada elemento del Product Backlog seleccionado, los Developers planifican el trabajo necesario para crear un Incremento que cumpla con la Definición de Terminado. A menudo, esto se hace descomponiendo los elementos del Product Backlog en elementos de trabajo más pequeños de un día o menos. La forma de hacerlo queda a criterio exclusivo de los

Developers. Nadie más les dice cómo convertir los elementos del Product Backlog en Incrementos de valor.

El Objetivo del Sprint, los elementos del Product Backlog seleccionados para el Sprint, más el plan para entregarlos se denominan juntos Sprint Backlog. La Sprint Planning tiene un límite de tiempo de máximo ocho horas para un Sprint de un mes. Para Sprints más cortos, el evento suele ser de menor duración.

Daily Scrum. El propósito de la Daily Scrum es inspeccionar el progreso hacia el Objetivo del Sprint y adaptar el Sprint Backlog según sea necesario, ajustando el trabajo planificado entrante.

La Daily Scrum es un evento de 15 minutos para los Developers del Scrum Team. Para reducir la complejidad, se lleva a cabo a la misma hora y en el mismo lugar todos los días hábiles del Sprint. Si el Product Owner o Scrum Master están trabajando activamente en elementos del Sprint Backlog, participan como Developers.

Los Developers pueden seleccionar la estructura y las técnicas que deseen, siempre que su Daily Scrum se centre en el progreso hacia el Objetivo del Sprint y produzca un plan viable para el siguiente día de trabajo. Esto crea enfoque y mejora la autogestión.

Las Daily Scrums mejoran la comunicación, identifican impedimentos, promueven la toma rápida de decisiones y, en consecuencia, eliminan la necesidad de otras reuniones. La Daily Scrum no es el único momento en el que los Developers pueden ajustar su plan. A menudo se reúnen durante el día para discusiones más detalladas sobre cómo adaptar o volver a planificar el resto del trabajo del Sprint.

Sprint Review. El propósito de la Sprint Review es inspeccionar el resultado del Sprint y determinar futuras adaptaciones. El Scrum Team presenta los resultados de su trabajo a los interesados clave y se discute el progreso hacia el Objetivo del Producto.

Durante el evento, el Scrum Team y los interesados revisan lo que se logró en el Sprint y lo que ha cambiado en su entorno.

Con base en esta información, los asistentes colaboran sobre qué hacer a continuación. El Product Backlog también se puede ajustar para satisfacer nuevas oportunidades. La Sprint Review es una sesión de trabajo y el Scrum Team debe evitar limitarla a una presentación.

La Sprint Review es el penúltimo evento del Sprint y tiene un límite de tiempo de máximo cuatro horas para un Sprint de un mes. Para Sprints más cortos, el evento suele ser de menor duración.

Sprint Retrospective. El propósito de la Sprint Retrospective es planificar formas de aumentar la calidad y la efectividad. El Scrum Team inspecciona cómo fue el último Sprint con respecto a las personas, las interacciones, los procesos, las herramientas y su definición de terminado. Los elementos inspeccionados suelen variar según el ámbito del trabajo. Se identifican los supuestos que los llevaron por mal camino y se exploran sus orígenes.

El Scrum Team analiza qué salió bien durante el Sprint, qué problemas encontró y cómo se resolvieron (o no) esos problemas. El Scrum Team identifica los cambios más útiles para mejorar su efectividad. Las mejoras más impactantes se abordan lo antes posible. Incluso se pueden agregar al Sprint Backlog para el próximo Sprint. La Sprint Retrospective concluye el Sprint.

Tiene un tiempo limitado a máximo tres horas para un Sprint de un mes. Para Sprints más cortos, el evento suele ser de menor duración.

Sprint Backlog. El Sprint Backlog se compone del Objetivo del Sprint (por qué), el conjunto de elementos del Product Backlog seleccionados para el Sprint (qué), así como un plan de acción para entregar el Incremento (cómo).

El Sprint Backlog es un plan realizado por y para los Developers. Es una imagen muy visible y en tiempo real del trabajo que los Developers planean realizar durante el Sprint para lograr el Objetivo del Sprint. En consecuencia, el Sprint Backlog se actualiza a lo largo del Sprint a medida que se aprende más. Debe tener suficientes detalles para que puedan inspeccionar su progreso en la Daily Scrum

Increment. Un Increment o incremento es un peldaño concreto hacia el Objetivo del Producto. Cada Increment se suma a todos los Increments anteriores y se verifica minuciosamente, lo que garantiza que todos los Increments funcionen juntos.

Para proporcionar valor, el Increment debe ser utilizable. Se pueden crear múltiples Increments dentro de un Sprint. La suma de los Increments se presenta en la Sprint Review apoyando así el empirismo. Sin embargo, se puede entregar un Increment a los interesados antes del final del Sprint. La Sprint Review nunca debe considerarse una puerta para liberar valor. El trabajo no puede considerarse parte de un Increment a menos que cumpla con la Definición de Terminado.

Definición de Terminado. La Definición de Terminado es una descripción formal del estado del Incremento cuando cumple con las medidas de calidad requeridas para el producto. En el momento en que un elemento del Product Backlog cumple con la Definición de Terminado, nace un Incremento. La Definición de Terminado crea transparencia al brindar a todos un entendimiento compartido de qué trabajo se completó como parte del Incremento. Si un elemento del Product Backlog no cumple con la Definición de Terminado, no se puede publicar ni presentar en la Sprint Review. En su lugar, vuelve al Product Backlog para su consideración futura. Si la Definición de Terminado para un Incremento es parte de los estándares de la organización, todos los Scrum Teams deben seguirla como mínimo. Si no es un estándar organizacional, el Scrum Team debe crear una Definición de Terminado apropiada para el producto.

Los Developers deben adherirse a la Definición de Terminado. Si hay varios Scrum Teams trabajando juntos en un producto, deben definir y cumplir mutuamente la misma Definición de Terminado

CAPITULO 3. DISEÑO METODOLOGICO DEL PROYECTO

En el presente capítulo se define el diseño metodológico empleado en este proyecto detallando los planes y estrategias que guiaron todo el proceso de desarrollo que se propuso.

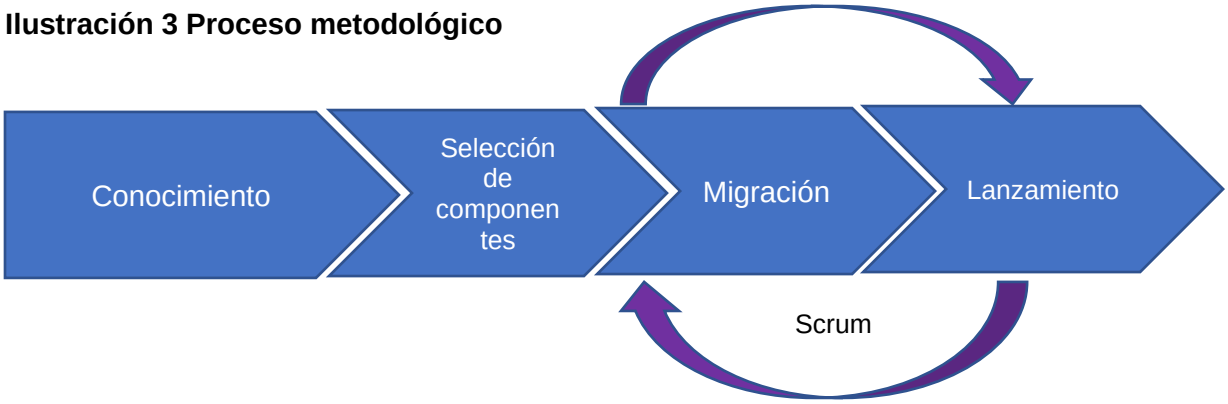
3.1. TIPO Y ENFOQUE DE LA INVESTIGACIÓN

El tipo de investigación seleccionado es descriptivo porque el propósito es mostrar con un nivel de detalle técnico todas las actividades realizadas para cumplir con los objetivos. El enfoque es cualitativo ya que no se harán mediciones específicas si no que se tratarán aspectos relativos a la descripción del comportamiento de los componentes del sistema.

3.2. METODOLOGÍA

Hemos diseñado un proceso metodológico compuesto por cuatro fases que son: conocimiento, selección de componentes, migración y lanzamiento. Cada fase se detalla a continuación:

Ilustración 3 Proceso metodológico



3.2.1 FASES DEL PROCESO METODOLÓGICO APLICADO.

Fase de conocimiento. Actividades:

Conocimiento del sistema actual: por medio de reuniones con los stakeholders se obtiene el conocimiento necesario para comprender el funcionamiento del sistema actual, se identifican los actores con sus respectivas responsabilidades y se identifican

los problemas o situaciones en las cuales se debe mejorar el comportamiento del sistema actual.

Fase de selección de componentes. Actividades:

Una vez conocido el sistema se pasa a indagar a cerca de las herramientas de hardware y software necesarias para el proceso de migración. Aquí es importante determinar las necesidades técnicas del nuevo sistema para tomar las mejores decisiones acerca de la nueva plataforma a utilizar de tal manera que los requisitos no funcionales del sistema se logren satisfacer.

Fase de migración. Con los componentes bien definidos y el conocimiento de los actores, se procede a seleccionar la metodología mas adecuada para el proceso de migración. Esta metodología o marco de trabajo debe soportar las operaciones que se establecen de manera prioritaria para obtener el éxito en la migración. Para este caso se ha seleccionado el marco de trabajo scrum.

Fase de lanzamiento. Finalizado el proceso de migración se procede a hacer el lanzamiento en vivió de los resultados para validar su funcionalidad.

3.2.2 INSTRUMENTOS PARA LA RECOLECCIÓN DE LA INFORMACIÓN.

Con el fin de obtener una información completa y concreta de los procesos, se planean una serie de reuniones con los usuarios principales para conocer de ellos la información que conlleve a la comprensión del sistema y construir las historias de usuario.

CAPITULO 4. INGENIERIA DEL PROYECTO

Resultados de la aplicación del marco de trabajo scrum

En este capítulo se detalla la adaptación de la metodología en el desarrollo de la aplicación móvil, explicando la implementación de cada una de sus fases de acuerdo a las necesidades del equipo de desarrollo.

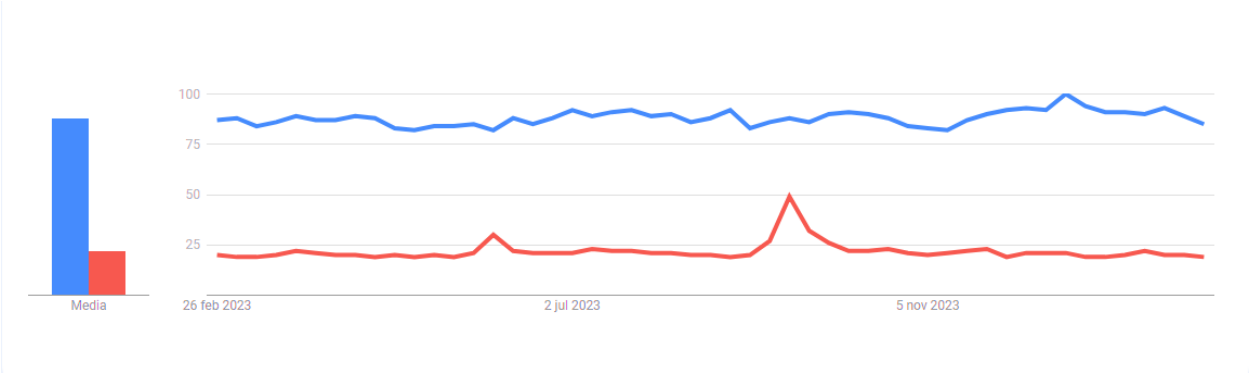
4.1. SITUACIÓN ACTUAL DE LA VERSIÓN ORIGINAL DEL APLICATIVO:

La aplicación actual desempeña un papel fundamental al proporcionar una plataforma que facilita la realización efectiva de charlas y talleres sobre el Green IT. Se trata de una aplicación nativa diseñada específicamente para el sistema operativo Android, lo que brinda la posibilidad de ser utilizada en una amplia variedad de dispositivos. Como se evidencia en la figura 2, la presencia del sistema operativo Android representado por la línea de color azul se destaca significativamente en los últimos años, abarcando dispositivos que van más allá de los móviles, incluyendo televisores, smartwatches, tablets y otros dispositivos. Cabe mencionar que el sistema operativo Android es, esencialmente, una variante del sistema operativo Linux, conocido por ser de código abierto. Esto posibilita que muchas empresas desarrolladoras o fabricantes de dispositivos utilicen este sistema operativo, con la excepción de casos como el de Huawei, donde restricciones políticas entre Estados Unidos y China llevaron a la prohibición de seguir comercializando dispositivos con Android.

Sin embargo, se presenta un desafío significativo al intentar captar a una audiencia que utiliza dispositivos iOS, dado que la aplicación actualmente se encuentra restringida exclusivamente a dispositivos con el sistema operativo Android. Esta restricción obstaculiza la participación de los usuarios de iOS en los talleres, dado que el uso de la aplicación como componente práctico resulta esencial para una correcta ejecución de las charlas impartidas. Ante este escenario, plantea la necesidad de

explorar soluciones que posibiliten una mayor accesibilidad y adaptabilidad de la aplicación a diversas plataformas.

Ilustración 4 Comparativa entre iOS y Android



4.2. PROCESO DE DESARROLLO BASADO EN SCRUM

4.2.1 CREACIÓN DEL EQUIPO SCRUM

La constitución del equipo de desarrollo juega un papel fundamental dentro del marco de trabajo Scrum, siendo un proceso que se extiende a lo largo de todo el ciclo de vida del proyecto y que no debe considerarse como una fase independiente. Durante este proceso, resulta crucial la selección de individuos idóneos y capacitados para ejecutar el proyecto, así como la definición clara de los roles y responsabilidades de cada miembro del equipo.

En el marco de trabajo Scrum, se enfatiza la importancia de mantener un tamaño de equipo reducido, ya que esto favorece la comunicación efectiva, la colaboración estrecha y la agilidad en la toma de decisiones. Un equipo más pequeño puede adaptarse rápidamente a los cambios, promoviendo así una ejecución más eficiente del proyecto y optimizando la interacción entre los miembros del equipo a lo largo de todo el desarrollo.

Para la ejecución del presente proyecto se presenta a continuación el equipo de desarrollo:

- **Product Owner**: de este rol se encargó la profesora Mary Luz Ojeda, conocedora del sistema y del proceso scrum.

- **Scrum Master**: este rol fue asumido por un desarrollador del proyecto con las habilidades necesarias para liderar, planificar y comunicarse efectivamente y de forma constantemente con el equipo para implementar de manera adecuada la metodología.

- **Development Team**: el equipo de desarrollo estuvo conformado por dos personas. Con conocimientos avanzados en desarrollo e infraestructura y con las capacidades necesarias para desarrollar tanto del lado del frontend como del backend, además de poder desplegar en un entorno final o de reproducción de la manera más eficiente.

Perfil de los miembros del equipo

Nombre	Mary Luz Ojeda Solarte
Rol	Product owner
Cargo	Ingeniera de sistemas
Perfil	

Nombre	Robinson Albeiro Tijo Gonzalez
Rol	Development team
Cargo	Desarrollador frontend
Perfil	Con más de dos años de experiencia como desarrollador frontend, se destaca por ser un experto en la creación de experiencias de usuario excepcionales mediante el uso de tecnologías como Angular, Flutter y Figma. Posee habilidades sólidas en el desarrollo de aplicaciones web y móviles, destacando su versatilidad en la creación de soluciones multiplataforma. A demás su capacidad para colaborar

	eficazmente en equipos ágiles, utilizando metodologías Scrum, ha sido fundamental para el éxito de múltiples proyectos en los que ha participado.

Nombre	Jaen Fellini Echavarria Tijo
Rol	Scrum master - development team
Cargo	Desarrollador backend
Perfil	Desarrollador backend con una sólida trayectoria de más de dos años, especializado en la implementación de soluciones robustas y eficientes. Experto en el uso de diversos lenguajes de programación, incluyendo Java, C# y Python. Comprometido con las buenas prácticas de programación y el mantenimiento de código limpio. Posee conocimientos avanzados en el diseño y desarrollo de microservicios para crear soluciones altamente escalables. Además, cuenta con habilidades en DevOps para realizar despliegues efectivos. Su experiencia en metodologías ágiles, especialmente en Scrum, lo capacita para desempeñar roles como Scrum Master.

4.2.2 PLANIFICACIÓN Y DEFINICIÓN DEL PROYECTO

Es en este punto donde el proyecto cobra forma a través de las historias de usuario redactadas por el PO (Product Owner). Estas historias de usuario se priorizan mediante una colaboración entre el equipo y los interesados, dando origen al Product Backlog que reflejará de manera precisa las necesidades críticas del proyecto. Además, en esta etapa se establece de manera definida el alcance del proyecto, delineando con claridad los límites y expectativas que guiarán su desarrollo.

En el ámbito de la gestión de proyectos es esencial contar con una herramienta que le permita al equipo realizar un seguimiento preciso del progreso del desarrollo. Contar con una plataforma que facilite el monitoreo detallado de las tareas, el avance de los hitos y la asignación de recursos no solo optimiza la eficiencia operativa, sino que también fomenta una comunicación transparente y una toma de decisiones informada dentro del equipo.

Existen diversas herramientas disponibles para la gestión de proyectos, entre ellas, Jira y Trello se posicionan como dos opciones populares y ampliamente utilizadas en el desarrollo de software, ambas herramientas son actualmente respaldadas y desarrolladas por la reconocida empresa Atlassian.

Jira y Trello comparten características similares como la colaboración en tiempo real y la capacidad de integración con diversas aplicaciones de terceros para ampliar sus funcionalidades. Sin embargo, se diferencian en su enfoque y complejidad. Trello destaca por su simplicidad y facilidad de uso al adoptar un sistema visual de tableros, listas y tarjetas. Aunque esta simplicidad lo hace intuitivo, también limita su capacidad para gestionar elementos más complejos de proyectos, como historias de usuario, epics y sprints, lo que lo hace ideal para proyectos más sencillos, pero menos adecuado para aquellos que requieren una gestión más detallada y estructurada. Por otro lado, Jira ofrece una mayor capacidad de personalización y funcionalidades avanzadas, convirtiéndolo en una opción más robusta para proyectos que demandan un enfoque más detallado y escalable en términos de gestión de tareas y procesos.

En nuestro caso, dado que el proyecto no tiene un tamaño extremadamente grande, Trello se ajusta de manera óptima a los requisitos específicos. Además, esta elección se ve respaldada por la ventaja adicional de que los miembros del equipo ya cuentan con experiencia previa en el uso de la herramienta, ya sea por su familiaridad o por haberla empleado en proyectos anteriores. Este conocimiento previo no solo facilita la transición hacia el uso de Trello, sino que también contribuye a una mayor eficiencia y comprensión del flujo de trabajo en el contexto de un proyecto de tamaño mediano.

En la siguiente ilustración, podemos apreciar la interfaz de la aplicación Trello, una herramienta esencial para la gestión de tareas en nuestro proyecto. La aplicación proporciona plantillas, o "templates" como suelen llamarlas, que permiten iniciar tableros con diseños prefabricados. No obstante, en este caso específico, hemos

optado por construir el tablero desde cero, lo cual implica la selección del tema o color del mismo, así como la creación de las diversas listas del tablero.

Dentro de estas listas, encontramos dos artefactos fundamentales de Scrum: el "product backlog" y el "sprint backlog". Además, otras listas como "in progress" y "done" desempeñan un papel crucial al proporcionar una trazabilidad de las tareas en curso y las que ya han sido completadas. La interfaz de Trello destaca por contar con una interfaz bastante limpia y altamente intuitiva, ofreciendo la posibilidad de potenciar sus capacidades con servicios de terceros, aunque en muchas ocasiones esta integración no resulta necesaria.

Es importante destacar, como se evidencia en la ilustración, que, al iniciar el proyecto, la única lista que debería estar completa con todas las tareas o historias de usuario por desarrollar es el "product backlog". No solo eso, sino que estas tareas ya deben estar organizadas por orden de prioridad de desarrollo.

Esta práctica inicial es fundamental para establecer una visión clara y jerarquía de las tareas pendientes. El "product backlog" sirve como un repositorio centralizado de todas las funcionalidades o mejoras planificadas para el proyecto, y ordenarlas según su prioridad proporciona una guía esencial para el equipo de desarrollo. Al seguir este enfoque, el equipo puede abordar las tareas en un orden estratégico, enfocándose en aquellas de mayor relevancia y valor para el proyecto en su conjunto.

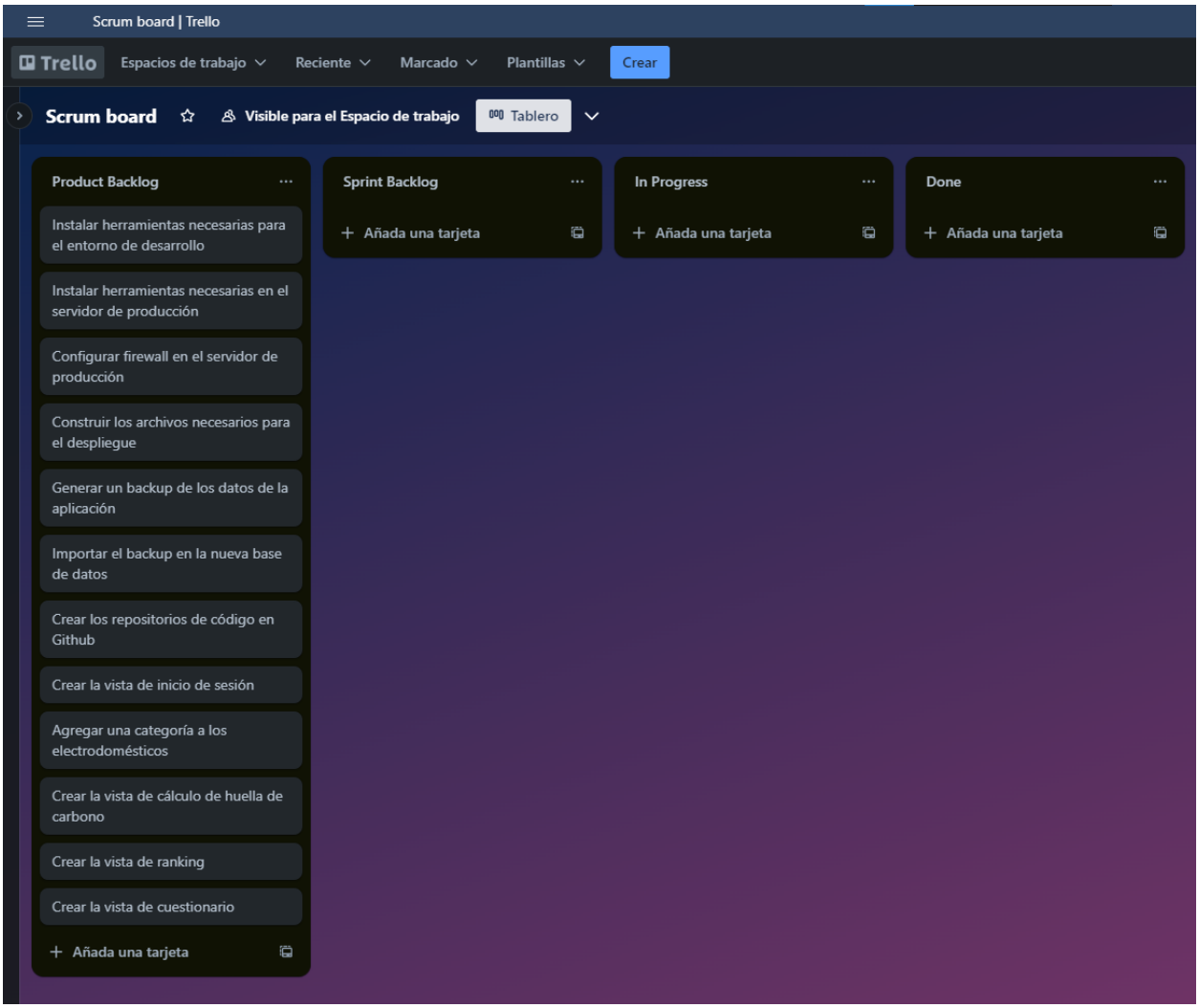
Además, mantener el "product backlog" como la única lista llena al inicio proporciona un panorama inicial claro y evita la dispersión de tareas en etapas tempranas. Esto contribuye a una planificación más efectiva y a la concentración en las actividades críticas al principio del proyecto, estableciendo así las bases para un desarrollo sistemático y centrado en los objetivos. En resumen, esta práctica inicial en la gestión del "product backlog" establece un sólido punto de partida para el equipo, permitiéndoles abordar las tareas de manera estructurada y alineada con los objetivos estratégicos del proyecto.

Una característica notable que Trello pone a nuestra disposición es la capacidad de asignar tareas a distintos miembros del equipo de manera eficiente. La interfaz se presenta extraordinariamente intuitiva, permitiendo a cualquier miembro del equipo, especialmente a los programadores, asignarse tareas con facilidad. Este proceso se lleva a cabo mediante la simple acción de mover la tarea desde la lista "sprint backlog" hacia "in progress". Este movimiento no solo indica visualmente que la tarea ha sido asignada, sino que también señala al resto del equipo que dicha tarea está ahora en desarrollo.

Una vez que la tarea ha sido completada y satisface todos los requisitos establecidos desde el principio en el "definition of done", procedemos a trasladarla hacia la lista "done". Este paso no solo marca el fin de la tarea, sino que también refleja la culminación exitosa de los procesos necesarios para considerarla como finalizada y lista para ser revisada o entregada.

Esta fluidez en la asignación, desarrollo y culminación de tareas se convierte en el núcleo del funcionamiento de la herramienta. Esencialmente, Trello se integra a la perfección en nuestro flujo de trabajo, brindando una visión clara y detallada del progreso de las tareas a lo largo del tiempo. La herramienta se convierte en una aliada valiosa para nuestro equipo de desarrollo, no solo simplificando la gestión de tareas, sino también facilitando la colaboración y el seguimiento transparente del avance del proyecto. La capacidad de asignación y el seguimiento visual eficaz de tareas, todo dentro de la interfaz intuitiva de Trello, contribuyen significativamente a la eficiencia y coordinación de nuestro equipo en cada etapa del desarrollo del proyecto.

Ilustración 5. Tablero scrum describiendo el product backlog



4.2.2.1 SPRINT 0.

Cuando se implementa la metodología ágil Scrum en equipos de desarrollo, es común dedicar un sprint completo a la preparación y adecuación de todo lo necesario para iniciar el desarrollo de las diversas historias de usuario. Este sprint inicial, conocido como "sprint 0", desempeña un papel esencial en la puesta en marcha del proyecto.

En el caso de nuestra implementación de la metodología ágil Scrum, dedicamos nuestro "sprint 0" a la cuidadosa selección y ejecución de tareas fundamentales que ocupan la parte inicial del "product backlog", como se detalla en la ilustración 5. Estas tareas, mayormente enfocadas en instalaciones y configuraciones, desempeñaron un papel clave al establecer las bases y condiciones necesarias para todo el entorno de desarrollo. Este enfoque estratégico nos permitió afrontar con éxito las demás tareas en los siguientes sprints.

El énfasis en las instalaciones y configuraciones en este sprint inicial es de vital importancia, ya que sienta los cimientos para el desarrollo continuo del proyecto en las etapas posteriores. Estas actividades abordaron necesidades críticas a nivel de infraestructura y configuración, asegurando que el equipo estuviera equipado con un entorno de desarrollo robusto y optimizado. Además, al situar estas tareas al inicio del "product backlog", garantizamos que la implementación inicial del proyecto esté respaldada por una base sólida.

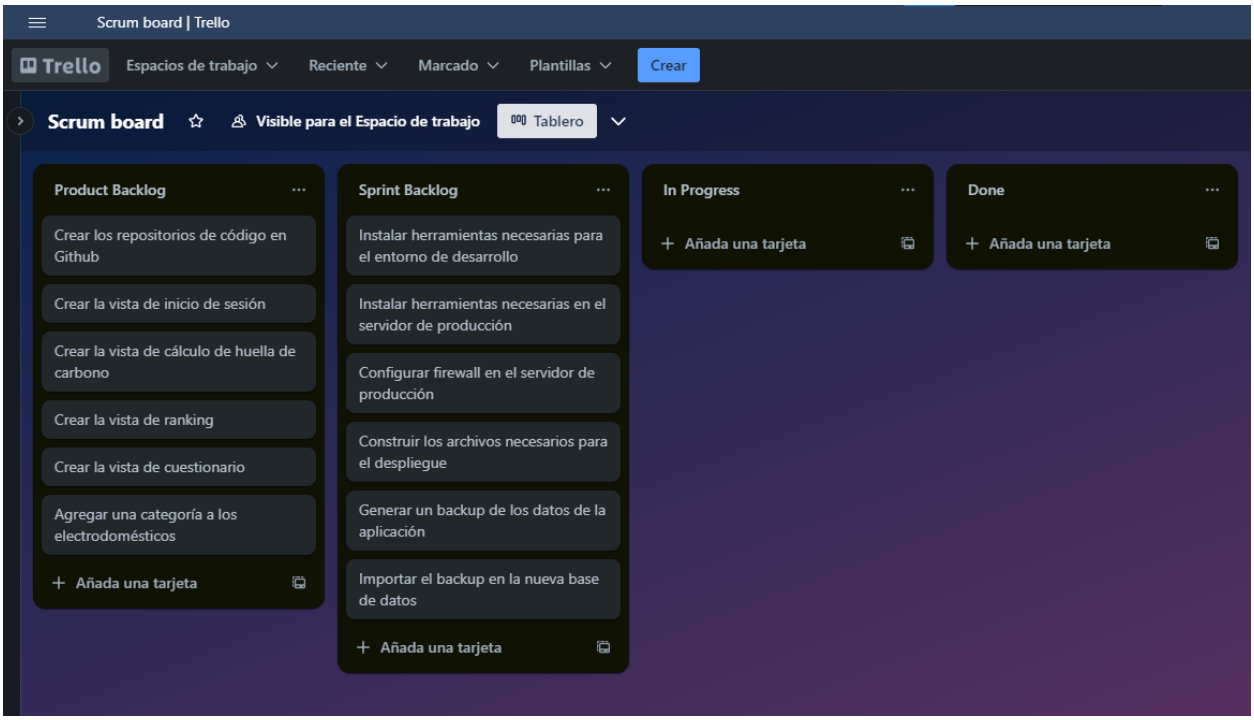
La ejecución exitosa de estas tareas iniciales no solo despejó el camino para los sprints posteriores, sino que también contribuyó significativamente a la eficiencia y eficacia del equipo de desarrollo. Al centrarnos en la preparación proactiva del entorno, pudimos minimizar posibles obstáculos y optimizar el flujo de trabajo del equipo en etapas críticas del proyecto. Este enfoque estratégico en el "sprint 0" demuestra su importancia como un precursor esencial para un desarrollo ágil y exitoso a lo largo del ciclo de vida del proyecto.

En la siguiente ilustración, se ofrece una visión detallada de las tareas que componen el "sprint 0". Este sprint no solo actúa como el punto de partida esencial para la preparación del proyecto, sino que también desempeña un papel crucial en la evaluación y medición de la capacidad de trabajo del equipo. La información recopilada durante este sprint inicial se convierte en un valioso indicador para realizar estimaciones más precisas en futuros sprints en relación con los tiempos de entrega y la carga de trabajo que el equipo puede gestionar con eficacia.

La ejecución del "sprint 0" proporciona insights valiosos sobre la dinámica del equipo y su capacidad para abordar tareas específicas. Este conocimiento adquirido se traduce directamente en mejoras sustanciales en la planificación y la asignación de recursos en los sprints posteriores. Al entender mejor la velocidad del equipo y sus fortalezas, podemos ajustar nuestras expectativas y planificaciones de manera más precisa, permitiendo una gestión más eficiente del proyecto en su conjunto.

En resumen, el "sprint 0" no solo es un precursor esencial para la fase de desarrollo, sino que también se convierte en una herramienta estratégica para perfeccionar la gestión del tiempo y la carga de trabajo del equipo. Esta evaluación inicial establece las bases para un proceso de desarrollo ágil y adaptativo, permitiendo al equipo abordar con confianza los desafíos futuros con una mayor comprensión de sus capacidades y eficiencia operativa.

Ilustración 6. Vista general del sprint “0”



4.3 ARQUITECTURA DEL SISTEMA

Al abordar la formulación de un proyecto o la implementación de cambios sustanciales en uno ya existente, resulta crucial plantearse la pregunta fundamental de cómo la integración de los diversos componentes a nivel global, como el frontend, backend, base de datos, servidor de correo, entre otros, podría verse afectada y cómo, en términos generales, estos elementos interactuarían entre sí. Al adoptar una perspectiva holística, se abre la puerta para explorar diversas soluciones que se adapten de manera óptima a las necesidades específicas del proyecto.

La consideración de la interdependencia entre los distintos elementos de un sistema es esencial para anticipar y gestionar posibles desafíos que puedan surgir durante la ejecución del proyecto. Evaluar cómo cada componente contribuye a un todo y cómo se relaciona con los demás proporciona una visión comprehensiva de la arquitectura general.

Esta visión global no solo facilita la identificación de posibles obstáculos o puntos de congestión, sino que también abre la puerta a la creatividad en la búsqueda de soluciones. Al comprender la forma en que estos componentes interactúan, es posible proponer estrategias y ajustes que optimicen la eficiencia y la cohesión del sistema en su conjunto.

Asimismo, tener una comprensión integral de la integración de componentes permite anticipar posibles necesidades de escalabilidad, garantizando que la solución propuesta no solo funcione eficientemente en el presente, sino que también sea flexible y adaptable a futuros cambios o incrementos en la carga de trabajo.

Comúnmente, una aplicación se estructura en dos componentes fundamentales: el frontend y el backend. El frontend, centrado en la parte visual dirigida al usuario, abarca diversas plataformas como aplicaciones móviles, aplicaciones de escritorio para diferentes sistemas operativos y aplicaciones web. En este ámbito, los desarrolladores

frontend suelen especializarse en áreas específicas, perfeccionando sus habilidades en un dominio particular. Aunque la especialización es común, se valora que estos profesionales tengan un conocimiento general del panorama para adaptarse a diversas situaciones.

En contraste, el desarrollador backend tiene como responsabilidad construir la lógica de negocio, tarea que anteriormente se realizaba en el lado del cliente. Con el tiempo, la programación se separó en el cliente y el servidor, dando origen a los desarrolladores frontend y backend. Antes de esta división, un único desarrollador se encargaba de ambos aspectos, conocido hoy como desarrollador fullstack.

Dentro de las tareas del desarrollador backend se encuentra la concepción y desarrollo de la API, junto con sus endpoints, que serán posteriormente consumidos por la aplicación frontend. Este proceso no solo implica la creación de la interfaz de programación de aplicaciones, sino también la cuidadosa definición de los puntos finales que facilitarán la interacción fluida entre el frontend y el backend.

Además, el desarrollador backend se encarga de la creación y configuración de la base de datos, siendo este un aspecto fundamental para el almacenamiento, recuperación y gestión eficiente de datos. Esta labor implica no solo el diseño inicial de la base de datos, sino también su conexión efectiva con la lógica de la aplicación y la garantía de su funcionamiento optimizado.

La configuración del servidor de correo es otra tarea esencial en la lista de responsabilidades del desarrollador backend. Esto implica asegurar que la aplicación pueda enviar y recibir correos electrónicos de manera eficiente y segura, aspecto crucial en numerosos contextos, desde la confirmación de cuentas hasta la gestión de notificaciones.

Adicionalmente, el desarrollador backend se enfrenta al desafío de integrar la aplicación con aplicaciones de terceros, como directorios activos o servicios de identidad. Esta

integración es vital para la seguridad y el proceso de inicio de sesión, ya que garantiza la autenticación eficaz y el acceso controlado a los recursos de la aplicación.

Un aspecto crucial es garantizar que la API esté siempre disponible para su consumo. Para lograrlo, el programador backend implementa estrategias para mejorar el rendimiento de la aplicación y garantizar su resiliencia ante posibles caídas. Asimismo, debe optimizar la aplicación sin alterar el comportamiento de la API y su integración con otros sistemas.

En entornos de alta demanda, las aplicaciones suelen enfrentar el desafío de manejar un tráfico considerable. En este escenario, una estrategia adoptada comúnmente es migrar a una arquitectura basada en microservicios. Aunque esta opción permite escalar la aplicación según se requiera, introduce una mayor complejidad en la gestión del sistema.

Las aplicaciones backend, conocidas como monolíticas, son las primeras que aprendemos al ingresar al mundo de la programación. Estas aplicaciones, por lo general, no requieren una complejidad significativa en su construcción. Son ideales para iniciar proyectos cuando no se tiene una idea clara del alcance desde el principio. Independientemente del lenguaje de programación utilizado, el concepto subyacente es el mismo: la aplicación backend expone una serie de endpoints utilizados por la aplicación frontend para la comunicación. Además, ejecuta la lógica de negocio y almacena los datos en una base de datos, ya sea relacional o no relacional. En algunos casos, se emplean herramientas como ORMs para facilitar la interacción con la base de datos, aunque no se consideran estrictamente necesarias. Aunque una aplicación backend monolítica puede incluir más componentes, en términos generales, así es como funciona.

En el contexto de este trabajo, la aplicación backend que amplía las funcionalidades se clasifica como monolítica. Esto aporta ciertas ventajas y desventajas, pero nos centraremos en una ventaja crucial: el mantenimiento. A diferencia de las aplicaciones

basadas en microservicios, que consisten en varias aplicaciones más pequeñas con bases de datos separadas, el mantenimiento de una aplicación monolítica es mucho más sencillo. Se simplifican los cambios necesarios, ya que se realizan en un solo lugar. Claro está que esto depende del diseño y tamaño de la aplicación, pero, en resumen, resulta más fácil realizar ajustes en una aplicación monolítica cuando sea necesario.

La manera en que las aplicaciones frontend y backend se comunican puede variar, y esto depende en gran medida de la construcción de la API. En el mercado, se encuentran tres enfoques comunes: SOAP, REST y GraphQL. Sin embargo, es el uso de REST el que predomina, principalmente debido a su capacidad para representar mensajes en el formato JSON, ampliamente aceptado y utilizado en el ámbito de la programación. REST se ha ganado la preferencia de los desarrolladores por su simplicidad, potencia y facilidad de comprensión, en contraste con el formato XML utilizado por SOAP en sus mensajes. Con REST, podemos aprovechar los métodos del protocolo HTTP, como POST, PUT, DELETE y GET, que se asocian comúnmente con las operaciones CRUD (Crear, Leer, Actualizar, Eliminar).

Este enfoque basado en REST no solo simplifica la implementación, sino que también se alinea con las prácticas convencionales de desarrollo, permitiendo a los desarrolladores utilizar métodos familiares y reconocidos para llevar a cabo operaciones esenciales en la aplicación. La elección de REST, respaldada por su uso de JSON y su integración intuitiva con las operaciones CRUD, contribuye a una comunicación más eficiente y coherente entre las aplicaciones frontend y backend.

Para nuestro caso en particular, la aplicación backend se basa en el uso de REST. La elección de este enfoque no fue complicada, considerando que el uso de REST se ha consolidado como un estándar en la industria, gracias a las ventajas previamente mencionadas. Además de definir el enfoque de comunicación, otro aspecto crucial es la selección de la librería que facilitará el envío de peticiones desde la parte frontend del sistema hacia el servidor o aplicación backend.

Aunque Angular ya integra una librería por defecto, la decisión de optar por Axios fue respaldada por la experiencia previa y el conocimiento acumulado en trabajos anteriores. La versatilidad y confiabilidad de Axios en situaciones similares la convirtieron en la elección ideal para gestionar las interacciones entre la aplicación frontend y la aplicación backend en este proyecto específico.

La elección de la librería adecuada para la comunicación entre el frontend y el backend es crucial, ya que afecta directamente la eficiencia y la robustez del sistema en su conjunto. Al basarnos en la experiencia previa con Axios, garantizamos no solo la familiaridad del equipo con la librería, sino también su capacidad comprobada para manejar de manera efectiva las solicitudes y respuestas, contribuyendo así a la fluidez y confiabilidad de la comunicación entre las distintas capas de la aplicación.

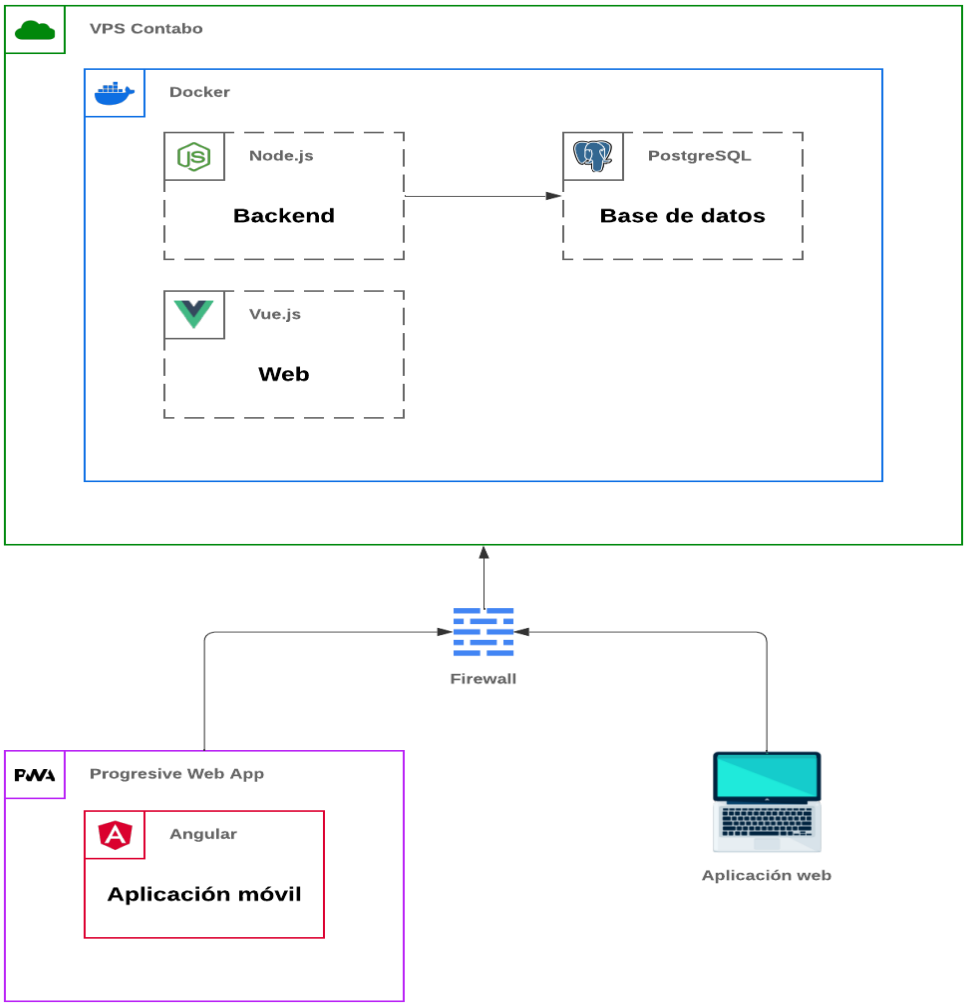
En la constante búsqueda de optimización y eficiencia, tomar decisiones informadas sobre las herramientas y tecnologías utilizadas es fundamental. En este caso, la combinación de REST como enfoque de comunicación y Axios como librería para facilitar dicha comunicación no solo se alinea con las prácticas de la industria, sino que también se adapta a las necesidades específicas del proyecto.

Una vez que hemos definido de manera general los componentes clave de la aplicación, estamos preparados para visualizarla como una sinfonía de engranajes que colaboran en perfecta sinergia para garantizar su funcionamiento integral. La Figura 7 nos proporciona una panorámica global de cómo el sistema se divide en dos partes fundamentales. Una de estas partes reside en el servidor, específicamente en un VPS (Servidor Privado Virtual), una máquina virtual contratada a través de una empresa de servicios en la nube, en nuestro caso, Contabo. La otra parte abarca las aplicaciones que los usuarios instalan en sus dispositivos, como las aplicaciones móviles, así como la aplicación web, accesible mediante un navegador. Aunque estas aplicaciones se ejecutan de forma descentralizada en los dispositivos de los usuarios, en última instancia, están alojadas en el servidor y se accede a ellas desde los navegadores web.

Dentro del servidor, observamos una serie de elementos cruciales. En primer plano, se destaca el firewall, actuando como una barrera de seguridad vital que gestiona las solicitudes entrantes. Esta capa de seguridad es esencial para mantener la integridad del sistema al limitar la exposición de puertos riesgosos, permitiendo únicamente aquellos que son esenciales para el funcionamiento del sistema.

Además, tanto la página web como la aplicación backend y la base de datos están alojadas y encapsuladas en contenedores Docker, una tecnología ampliamente utilizada en la industria del software. Estos contenedores proporcionan un entorno aislado y consistente, asegurando que las aplicaciones y la base de datos se ejecuten de manera eficiente y predecible, independientemente del entorno en el que se desplieguen.

Ilustración 7. Diagrama de componentes del sistema


VPS Contabo
Docker
Node.js
Backend
PostgreSQL
Base de datos
Vue.js
Web
Firewall
Progresive Web App
Angular
Aplicación móvil
Aplicación web

.

4.4 DESARROLLO DE LA APLICACIÓN MÓVIL

Tras identificar el desafío de no contar con una aplicación móvil multiplataforma, lo cual limitaba la captación de un segmento de usuarios que utilizaba dispositivos con sistema operativo iOS, se emprendió la búsqueda de soluciones que ofrecía el mercado. Entre las alternativas consideradas se destacaron las siguientes:

Desarrollo de Aplicación Nativa para iOS: La opción de crear una aplicación específicamente diseñada para iOS, utilizando lenguajes como Swift o Kotlin, fue contemplada. Esta estrategia implicaría la creación de dos versiones separadas de la aplicación, una para iOS y otra para Android.

Adopción de Flutter para Desarrollo Multiplataforma: Otra alternativa evaluada consistió en rediseñar la aplicación móvil utilizando Flutter. Esta opción permitiría desarrollar una única base de código que funcionaría tanto en dispositivos iOS como en Android, facilitando la gestión y mantenimiento de la aplicación para ambas plataformas.

Implementación de Progressive Web App (PWA): La tercera opción considerada fue la reconstrucción de la aplicación móvil como Progressive Web App (PWA). Esta elección permitiría ejecutar la aplicación de manera independiente del dispositivo y sistema operativo, brindando una experiencia de usuario consistente a través de diferentes plataformas.

Cada una de las soluciones contempladas presenta ventajas y consideraciones específicas. La elección entre ellas fue influenciada por factores cruciales como la audiencia objetivo y, sobre todo, los recursos disponibles. Las características particulares del proyecto desempeñaron un papel determinante en la decisión de optar por una solución específica, siendo la creación de una Progressive Web App (PWA) la opción seleccionada.

Una Progressive Web App (PWA), en esencia, se presenta como una página web empaquetada que brinda la experiencia de utilizar una aplicación móvil. Este enfoque permite a los desarrolladores aprovechar beneficios característicos de las aplicaciones móviles, como notificaciones push, acceso a la cámara, ubicación y otros sensores del

dispositivo. Aunque no puede compararse con la capacidad de controlar diversos sensores como una aplicación nativa, resulta una excelente elección para desarrollar un Producto Mínimo Viable (MVP) o una aplicación que no requiera un control exhaustivo del hardware del dispositivo.

En el proceso de creación de una PWA, se dispone de herramientas como Angular, React y Vue, cada una ofreciendo resultados similares. La elección entre estas herramientas no varía significativamente en su uso, siendo la experiencia previa del desarrollador el factor clave. Estas herramientas comparten conceptos esenciales como la programación reactiva y la reutilización de componentes. Además, tanto Angular como React y Vue cuentan con comunidades robustas y una amplia variedad de librerías para diversas funcionalidades. Todas ellas admiten tanto Javascript como Typescript como lenguajes de programación.

La decisión de optar por Angular se basó, en primer lugar, en la experiencia previa del desarrollador frontend. En segundo lugar, Angular destaca por ser un framework con una estructura claramente definida, lo que agiliza la toma de decisiones sobre la creación y ubicación de diferentes componentes de la página web. También ofrece una forma intuitiva y sencilla de empaquetar la aplicación final en el service worker.

La toma de decisiones fundamentales al iniciar un proyecto incluye la definición temprana de la estructura organizacional y la arquitectura que se empleará. Este enfoque garantiza que los desarrolladores sigan un estilo de organización coherente y una metodología unificada para abordar cada problema. Tal cohesión no solo mejora la comunicación dentro del equipo, sino que también facilita la integración fluida de nuevos miembros en el futuro.

En nuestro caso, tras evaluar diversas arquitecturas disponibles, como la tradicional Modelo-Vista-Controlador (MVC), la emergente pero resonante Micro Frontends, la basada en componentes, y la Atomic Design, hemos optado por la arquitectura basada en componentes. Esta elección se fundamenta en sus ventajas, como mantener toda la

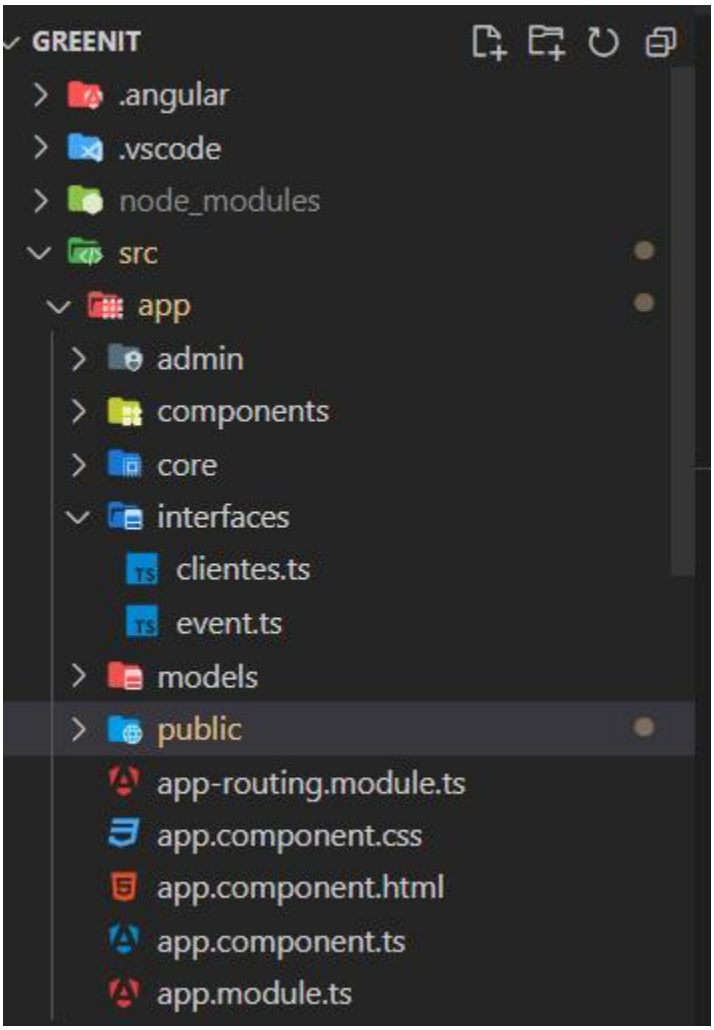
lógica de negocio desacoplada entre sí y la capacidad de reutilizar funcionalidades de manera orgánica. Además, su implementación resulta sencilla, contribuyendo a una estructura de proyecto notablemente organizada.

La siguiente ilustración presenta una visión global de la estructura organizacional del proyecto. En este contexto, Angular desempeña un papel fundamental al proporcionar un proyecto esqueleto preconfigurado al iniciar un nuevo proyecto. Este proyecto esqueleto sirve como una base sólida desde la cual se puede avanzar en el diseño y desarrollo de la aplicación.

Al crear un nuevo proyecto en Angular, la plataforma no solo suministra una estructura básica, sino que también ofrece todos los elementos esenciales necesarios para que la aplicación sea operativa y funcional. Aunque la aplicación base proporcionada puede considerarse elemental en su complejidad, está completa con todas las características y recursos necesarios para su funcionamiento efectivo.

En este sentido, el enfoque de Angular al proporcionar un punto de partida estructurado permite a los desarrolladores ahorrar tiempo en la configuración inicial del proyecto y enfocarse directamente en la implementación de la lógica de la aplicación. Esta capacidad de Angular para facilitar el inicio rápido del desarrollo y brindar una base sólida contribuye significativamente a la eficiencia y la eficacia en el proceso de creación de aplicaciones.

Ilustración 8. Vista global de los componentes del proyecto

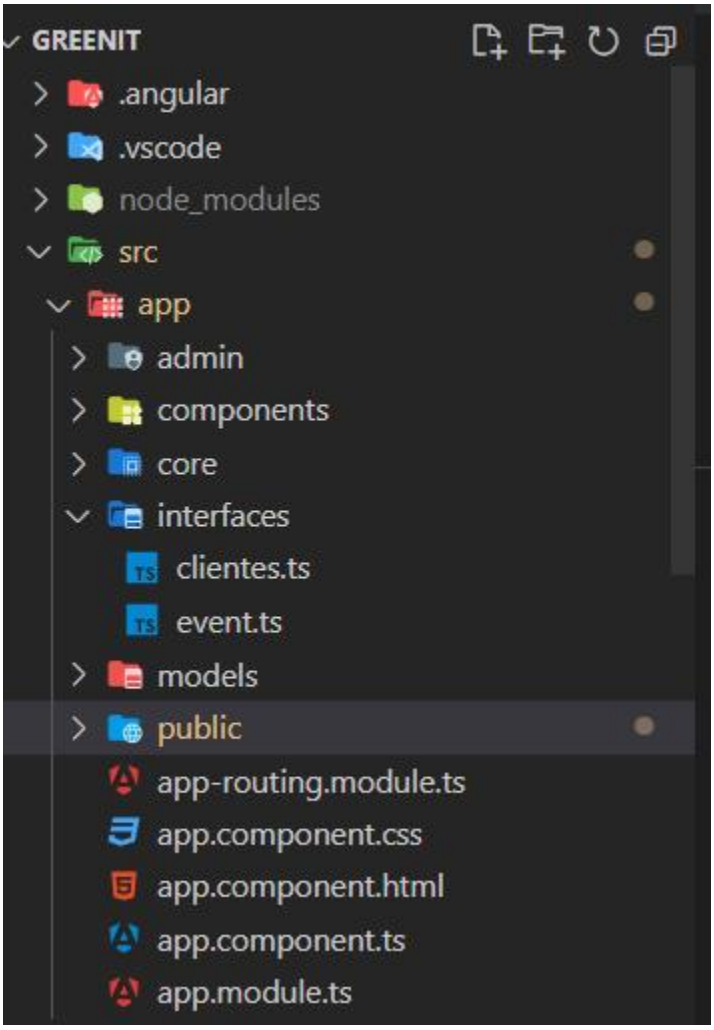


Al explorar la carpeta "public" después de haber implementado la arquitectura en la aplicación, se presenta una visión detallada de las diversas pantallas o vistas que constituyen la aplicación, como se ilustra en la ilustración 8. Al mismo tiempo, dentro de la carpeta "interfaces", también visible en la ilustración 8, nos encontramos con dos archivos que llevan la extensión ".ts", indicando claramente que se trata de archivos TypeScript. Estos dos archivos desempeñan un papel fundamental al representar los datos provenientes del backend como objetos. En otras palabras, estas estructuras de datos no solo sirven como componentes fundamentales, sino que también actúan como cimientos cruciales que otros componentes utilizan para desarrollar la lógica de negocio de la aplicación.

Es importante resaltar que TypeScript fue concebido por Microsoft con el propósito de potenciar las capacidades de JavaScript, un lenguaje ya establecido y ampliamente utilizado en el desarrollo web. La mejora principal que aporta TypeScript es la introducción de tipado estático, lo que añade la capacidad de detectar errores en tiempo de compilación, o más específicamente, durante el proceso de transpilación. En este proceso, el código TypeScript se traduce a código JavaScript completamente ejecutable.

Además de la mejora en la detección de errores, TypeScript ofrece ventajas notables sobre JavaScript en términos de programación orientada a objetos. Facilita la colaboración entre equipos al proporcionar mayor claridad en el código gracias al tipado estático, lo que es crucial en proyectos de gran escala. Asimismo, contribuye a optimizar la refactorización del código al permitir realizar cambios con mayor seguridad, ya que estos se pueden verificar y validar durante la fase de compilación. En última instancia, las características añadidas por TypeScript aportan no solo a la robustez y mantenimiento del código, sino también a la eficiencia en el desarrollo de aplicaciones, brindando un conjunto robusto de herramientas para enfrentar los retos de proyectos modernos y complejos.

Ilustración 9. Vista ampliada de la carpeta “public”



Si examinamos el contenido de cada pantalla que constituye la aplicación, nos encontraremos con varios archivos que desempeñan funciones específicas. Algunas bibliotecas, como React, proponen encapsular varias responsabilidades dentro de un solo archivo. Este enfoque ha generado tanto elogios como críticas por parte de desarrolladores en todo el mundo. Aunque la ventaja evidente es la cohesión de todas las funciones en un solo archivo, lo que facilita la modificación al tener todo en un mismo lugar, a largo plazo, desde nuestra perspectiva, puede convertirse en una fuente de frustración. Esto se debe a que los archivos tienden a crecer desmesuradamente, y como desarrolladores, nos enfrentamos al desafío de perder fácilmente el hilo al intentar leer el código y comprender su funcionalidad o comportamiento.

En este sentido, la práctica de consolidar diversas responsabilidades en un solo archivo, si bien puede simplificar la gestión inmediata, puede derivar en una falta de mantenibilidad y claridad a medida que el proyecto evoluciona. Es crucial sopesar los beneficios inmediatos de la cohesión frente a las posibles complicaciones futuras, buscando un equilibrio que favorezca tanto la eficiencia inicial como la sostenibilidad a largo plazo del código.

Angular, como marco de trabajo o framework bien conocido, introduce una estructura organizativa que diferencia claramente los componentes esenciales de una aplicación. Este enfoque se evidencia en la separación de los archivos destinados al código HTML, CSS y TypeScript, encargados respectivamente de la representación estructural de la página web, la estilización de dicha estructura y la lógica de negocio asociada.

La filosofía de Angular, al mantener estos tres aspectos cruciales en archivos independientes, se alinea de manera coherente con el principio de responsabilidad única, uno de los pilares fundamentales de los principios SOLID en el diseño de software. Según este principio, cada archivo o componente debería tener una única razón para cambiar, evitando la acumulación de múltiples responsabilidades en un solo lugar.

Desde nuestra perspectiva, esta metodología de organización resulta altamente beneficiosa. Al asignar a cada archivo una tarea específica, se promueve la claridad y la mantenibilidad del código. Cada componente, ya sea el HTML, el CSS o el TypeScript, se convierte en una entidad autónoma que se puede entender y modificar de manera aislada, facilitando la colaboración entre desarrolladores y reduciendo la complejidad asociada con la gestión de archivos extensos.

Este enfoque también contribuye a la escalabilidad del proyecto, ya que permite un fácil mantenimiento y extensión de la aplicación sin comprometer la legibilidad del

código. Además, al adherirse a los principios SOLID, Angular ofrece una base sólida para el desarrollo de aplicaciones que pueden evolucionar de manera eficiente y sostenible a medida que los requisitos cambian y crecen con el tiempo. En resumen, la estructura propuesta por Angular, al enfocarse en la modularidad y la responsabilidad única, proporciona una sólida base para el desarrollo de aplicaciones robustas y mantenibles a largo plazo.

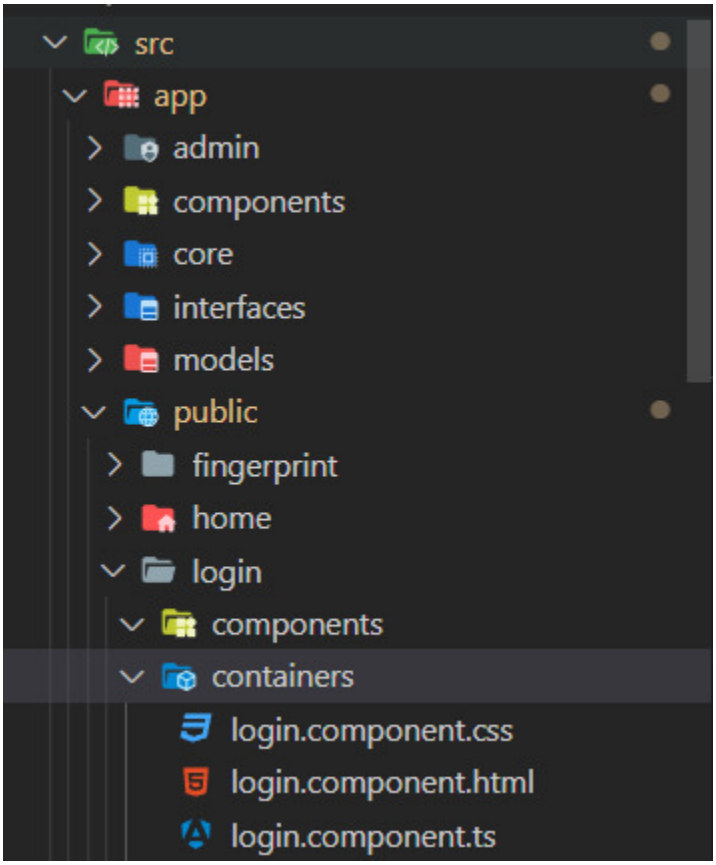
La ilustración 10 proporciona una visualización evidente de la estructura organizativa empleada para la pantalla de inicio de sesión. En esta representación, se observa claramente que los archivos HTML, CSS y TypeScript están estructurados de manera independiente, siguiendo una metodología de desarrollo que promueve la modularidad y la claridad en la gestión del código. No solo se limita a la separación de estos elementos fundamentales, sino que también introduce un directorio denominado "components", destinado específicamente a albergar componentes más pequeños que, en conjunto, conforman la totalidad de la vista o pantalla de inicio de sesión.

Este enfoque de modularidad no solo se traduce en una mayor legibilidad del código, sino que también facilita el mantenimiento y la expansión del proyecto. Cada archivo desempeña un rol específico y se puede modificar o mejorar de manera aislada, lo que simplifica el proceso de desarrollo y depuración. La adopción de un directorio exclusivo para componentes más pequeños amplía aún más la capacidad de reutilización y organización, permitiendo una gestión más eficiente de las piezas individuales que contribuyen al conjunto.

La adopción de esta estructura no solo se limita a la comodidad en el desarrollo, sino que también propicia la colaboración entre equipos, ya que cada miembro puede enfocarse en áreas específicas sin entorpecer el trabajo de los demás. Además, proporciona una base sólida para futuras iteraciones y actualizaciones, ya que la modularidad inherente facilita la introducción de nuevas funcionalidades o la adaptación a cambios en los requisitos del proyecto.

En definitiva, la disposición clara y modular de los archivos en la estructura de la pantalla de inicio de sesión no solo mejora la eficiencia en el desarrollo, sino que también sienta las bases para un mantenimiento sencillo y una evolución ágil del proyecto a medida que evolucionan las necesidades y exigencias.

Ilustración 10. Vista ampliada de la pantalla de inicio o login



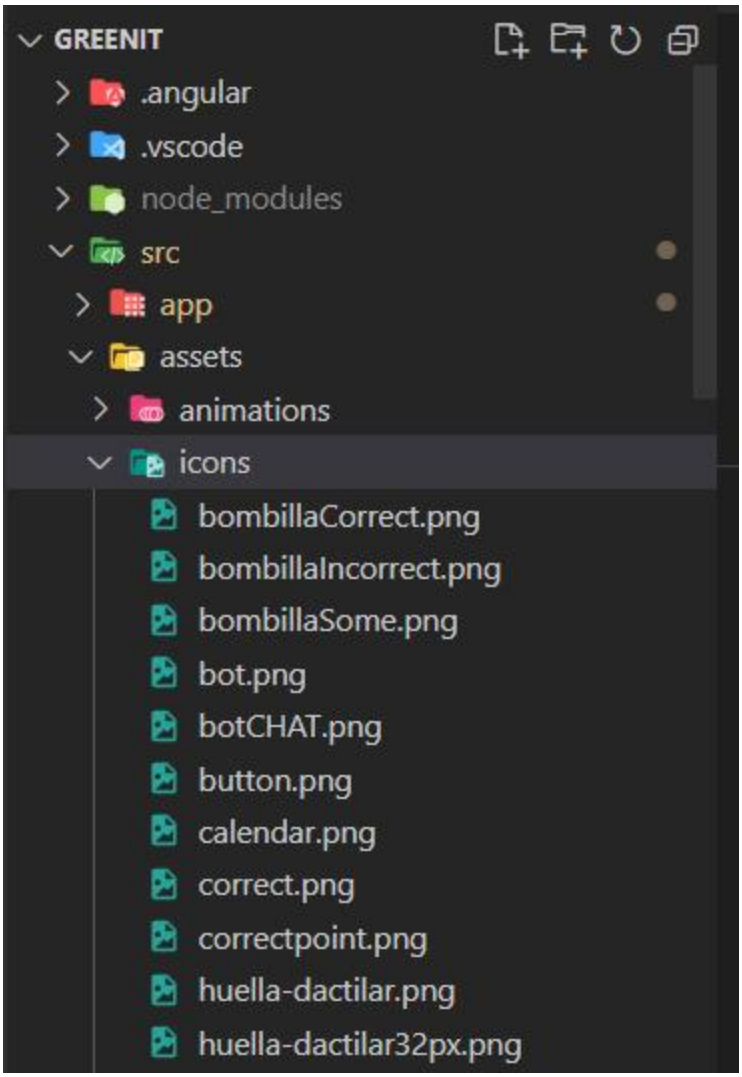
En la actualidad, la importancia de concebir interfaces de usuario que no solo sean funcionales, sino también intuitivas y estéticamente agradables, se ha vuelto crucial para enriquecer la experiencia del usuario. Con este objetivo en mente, se ha optado por la inclusión de diversos iconos y animaciones, ampliamente reconocidos como patrones de diseño que los usuarios suelen encontrar en otras aplicaciones. Esta elección no solo busca mejorar la estética visual, sino que también tiene como objetivo mejorar la usabilidad general de la aplicación.

En la Figura 6, se destaca la existencia de una carpeta específica denominada "assets", la cual está diseñada para almacenar una variedad de recursos esenciales. Estos recursos van desde imágenes hasta iconos, animaciones y fuentes, entre otros elementos visuales. Esta dedicación de un espacio exclusivo para tales recursos subraya la importancia atribuida a la estética y a la experiencia visual en el diseño de la aplicación.

La inclusión de iconos y animaciones reconocibles como patrones de diseño no solo cumple con el propósito estético, sino que también capitaliza la familiaridad de los usuarios con estos elementos. Al aprovechar patrones que los usuarios han encontrado en otras aplicaciones, se crea una sensación de coherencia y facilita la comprensión de la interfaz por parte del usuario, mejorando así la experiencia global.

La carpeta "assets" no solo actúa como un repositorio para imágenes y elementos visuales, sino que representa un componente esencial para la cohesión y consistencia del diseño. Facilita la gestión de recursos gráficos y garantiza que estos estén fácilmente accesibles para su implementación en diferentes partes de la aplicación. En última instancia, esta atención al detalle en la elección y organización de recursos visuales contribuye significativamente a la creación de una interfaz de usuario atractiva y funcional que cumple con los estándares modernos de usabilidad.

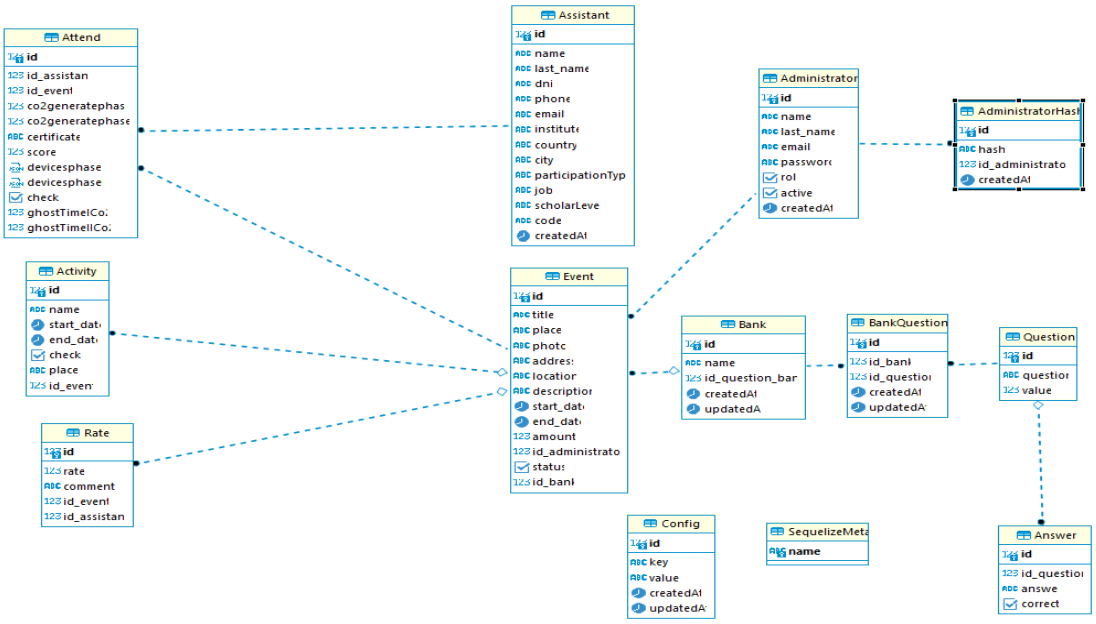
Ilustración 11. Vista ampliada de los recursos utilizados en la aplicación



4.5 MODELO DE LA BASE DE DATOS

La siguiente ilustra contiene el modelo relacional empleado para el aplicativo.

Ilustración 12. Modelo de la base de datos



4.5.1 DEFINICIÓN DE MODELO ENTIDAD RELACIÓN

Una base de datos entidad-relación es un modelo de datos que se utiliza para describir las entidades o conceptos relevantes en un dominio de aplicación y las relaciones entre ellas. En el contexto que mencionas, se hace referencia a una base de datos que probablemente esté relacionada con la gestión de eventos, asistentes, administradores, banco de preguntas que se relacionan con un banco de respuestas.

Aquí hay una explicación básica de las entidades que podrían estar presentes en la base de datos y sus relaciones:

- Event: Esta tabla representa los eventos que se organizan o gestionan. Cada registro en esta tabla contiene información como el nombre del evento, la fecha y la ubicación.
- Assistand: Esta tabla representa a los asistentes que participan en los eventos. Cada registro podría incluir detalles como el código de acceso, nombre del asistente, información de contacto, etc.

- Attend: Esta tabla es significativa ya que gestiona información como los puntos obtenidos por el usuario, el CO_2 generado, los dispositivos seleccionados, etc.
- Administrator: Esta tabla contiene información sobre los administradores que son responsables de la gestión de los eventos. incluye datos como el nombre del administrador, rol, permisos de acceso, etc.

4.6. PROTOTIPADO. DISEÑO DE LAS INTERFACES GRÁFICAS DE USUARIO FINAL

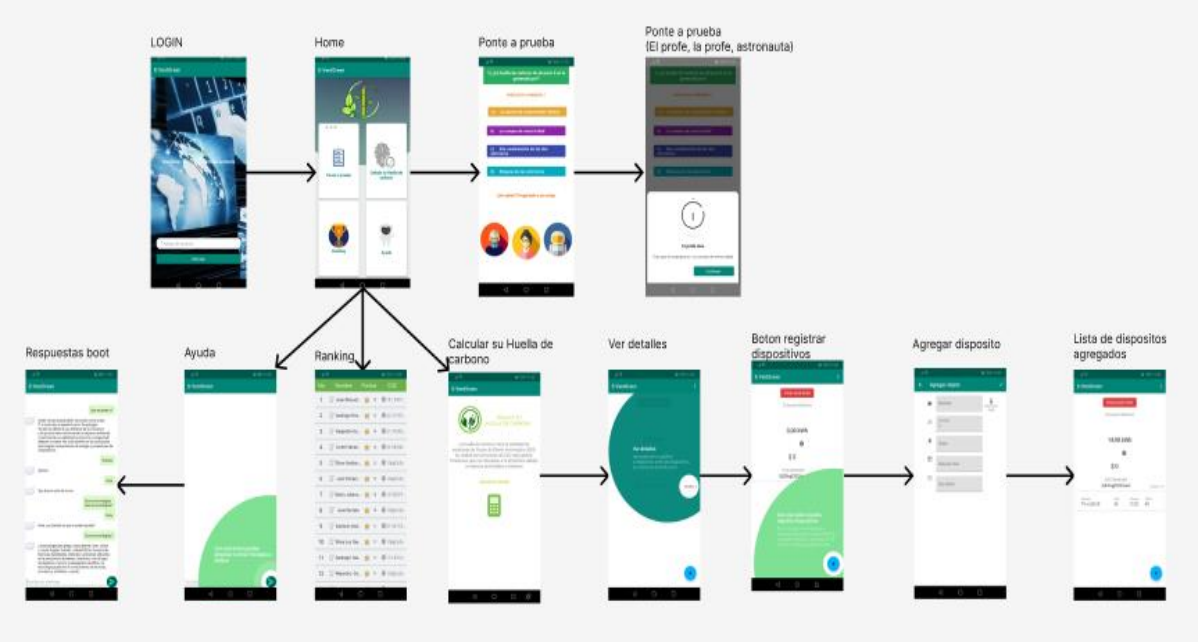
Se realiza prototipado de la aplicación para que se puedan validar términos de fidelidad y funcionalidad, con el fin de probar conceptos, diseños responsivos, ideas, recopilar información y guiar el desarrollo. Lo cual utilizamos una herramienta como Figma que nos ayuda al diseño y prototipado hay va razones por las cuales podría ser una opción preferida para desarrollar prototipos de aplicaciones:

- **Colaboración en tiempo real**: Figma permite a varios miembros del equipo trabajar en un diseño al mismo tiempo, lo que facilita la colaboración y la comunicación entre diseñadores, desarrolladores y otros.
- **Plataforma basada en la nube**: Al ser una aplicación basada en la nube, Figma elimina la necesidad de preocuparse por versiones desactualizadas o problemas de compatibilidad. Los diseños se guardan automáticamente y están siempre disponibles para acceder desde cualquier lugar y en cualquier momento.
- **Prototipado interactivo**: Figma permite crear prototipos interactivos directamente en la misma plataforma de diseño, lo que facilita la transición del diseño estático a una experiencia de usuario más dinámica. Los prototipos pueden incluir transiciones, enlaces y elementos interactivos que simulan el comportamiento de la aplicación final.
- **Facilidad de uso**: Figma tiene una interfaz intuitiva y fácil de usar que facilita tanto a principiantes como a profesionales trabajar en el diseño y prototipado de aplicaciones. Además, ofrece una amplia gama de recursos de aprendizaje,

incluyendo tutoriales y documentación, para ayudar a los usuarios a aprovechar al máximo la herramienta.

- **Integraciones**: Figma se integra con una variedad de herramientas y servicios, como Slack, Jira y Zeplin, lo que facilita la colaboración y la transferencia de diseños a otras plataformas utilizadas en el proceso de desarrollo.

Ilustración 13. Interfaces gráficas de usuario



4.6.1. DESCRIPCIÓN DE DISEÑO.

Las siguientes ilustraciones muestran las interfaces gráficas de usuario final:

- ilustración 14: interfaz de acceso, es la pantalla que permite a los usuarios contemplar un video interactivo enfocado en el Green it, plasmando un campo de código autenticación.
    - Ilustración 15 interfaz notificación de acceso, se visualizará una notificación el cual nos informa de novedades que se estén presentando al momento de ingresar.
    - Ilustración 16: interfaz de inicio, en este apartado se evidenciará 4 menús de navegación, cada uno con iconos interactivos que ayudan a la exploración de los usuarios.

- Ilustración 17: interfaz gráfica calcular huella, allí se encontrará una breve definición de Huella de carbono, también observaremos un menú desplegable el cual nos servirá para calcular la huella de carbono del entorno que seleccionemos.
- Ilustración 18: interfaz gráfica cuestionario, permitirá al usuario responder preguntas referentes al greenit, también encontraremos instrucciones y ayudas intuitivas para fortalecer el conocimiento en los usuarios.
- Ilustración 19: interfaz gráfica de ranking, se presenta información de manera clara y efectiva, mediante una grilla que compara la información suministrada en los módulos de cuestionario y calcula tu huella de carbono.
- Ilustración 20: interfaz gráfica de chat, el usuario realizara preguntas relacionadas con el green it, también encontrara un área conversacional el cual es el historial de las preguntas que le realizo a Bot.

Ilustración 14. Interfaz de acceso

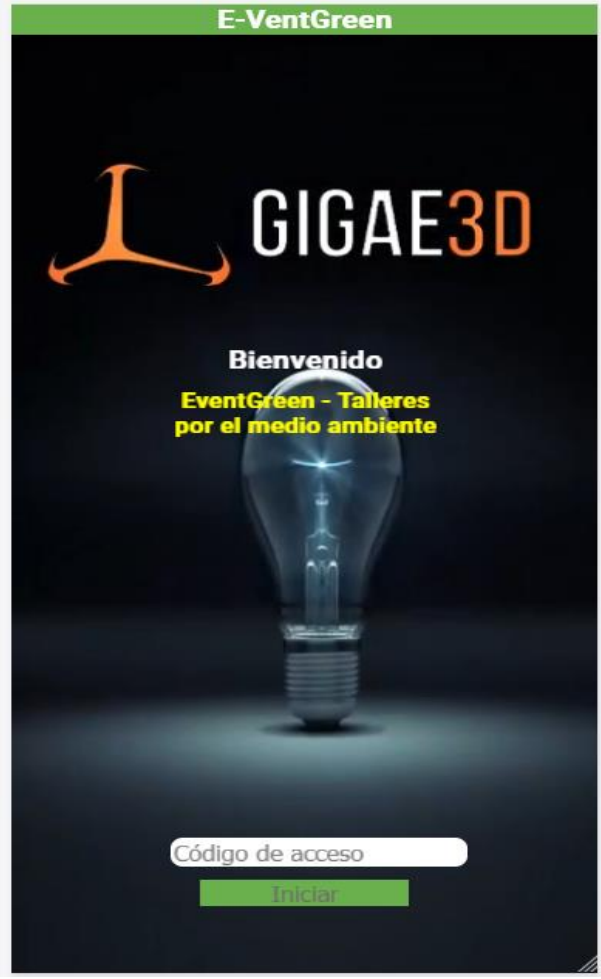


Ilustración 15. Interfaz gráfica notificación de acceso.

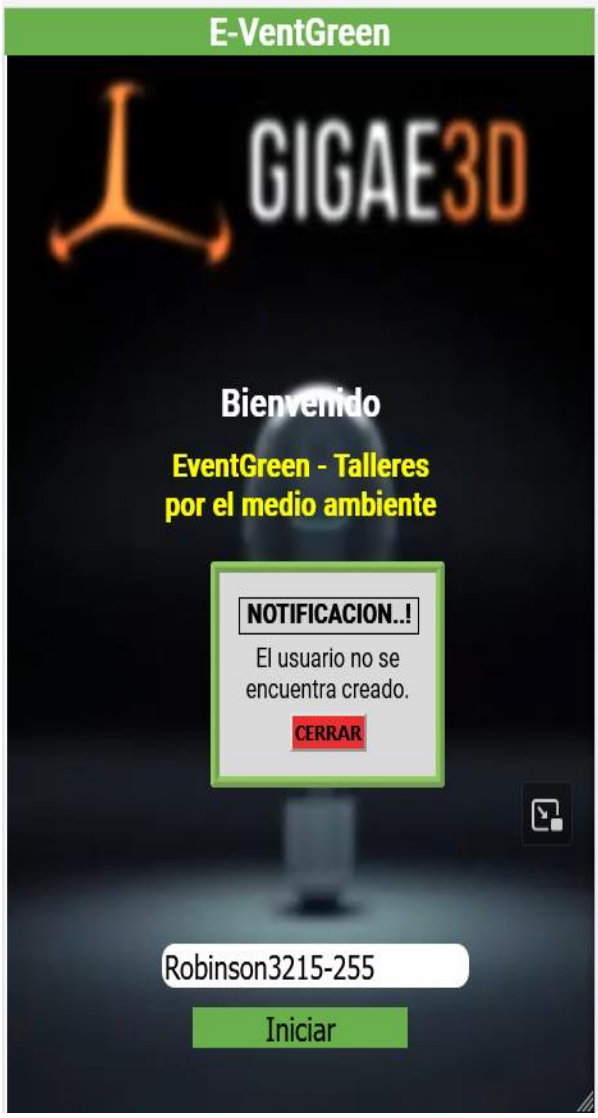


Ilustración 16. Ilustración 3. Interfaz gráfica de inicio.

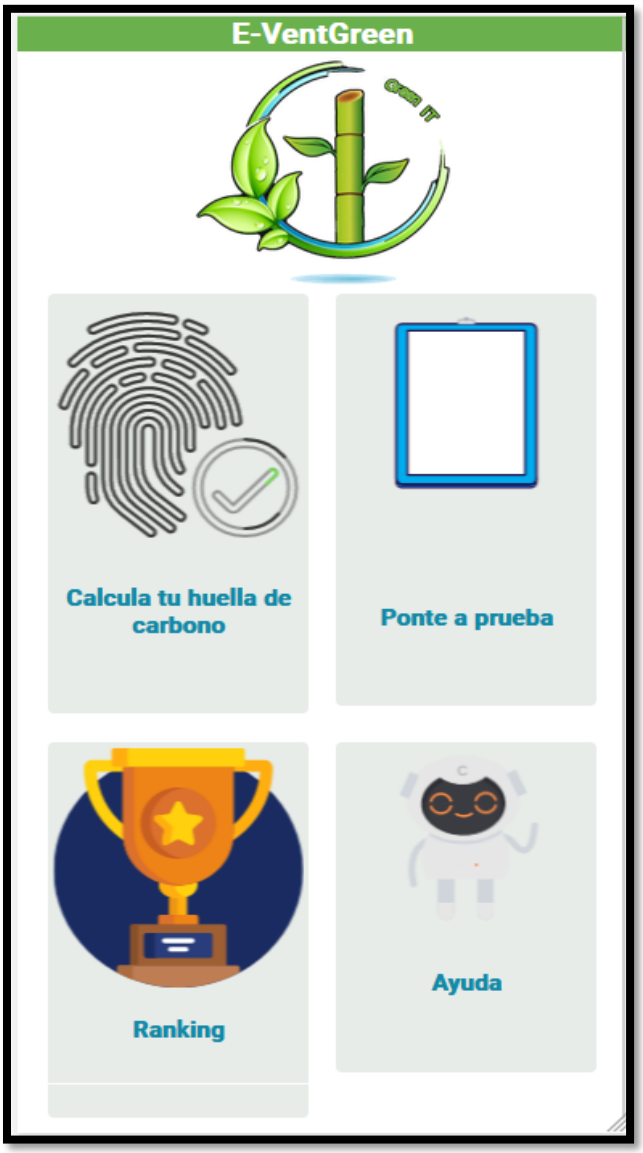


Ilustración 17. . Interfaz gráfica calcular huella

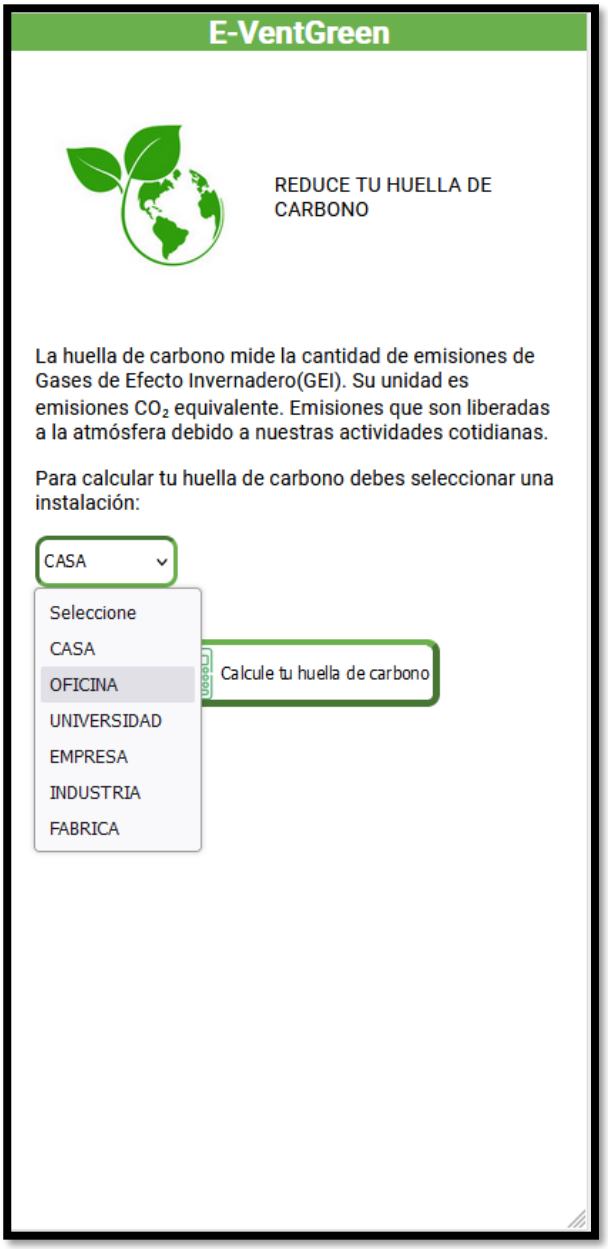
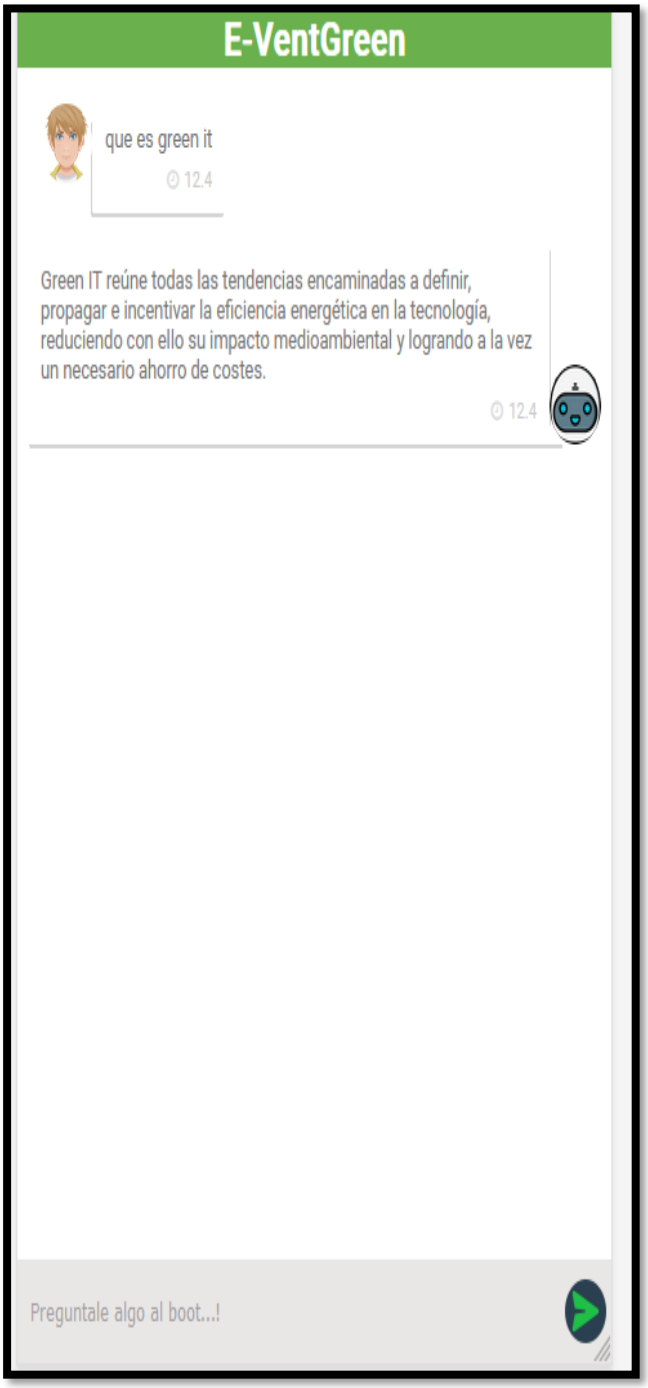


Ilustración 18. Interfaz gráfica cuestionario.

Ilustración 19. Interfaz gráfica ranking

E-VentGreen			
No	Nombres	Puntos	C02
1	Juan Fernando Marín Moncada	12	0.000000
2	Elinor Nathaniel Murillo Constain	12	0.000000
3	María Juliana Ruiz Arias	12	13.926193
4	Alejandro Ocampo Lopez	12	51.375461
5	Jose Alejandro Hernandez Vallecilla	11	153.893802
6	Alejandro Gonzalez Rico	11	0.000000
7	Juan Jose Blandón González	10	0.411499
8	Juan Pablo González Velasco	9	134.008653
9	Hector Felipe Copete Vallecilla	9	0.000000
10	Alejandra María Rodríguez Correa	9	157.379320

Ilustración 20. Interfaz gráfica de chat



4.4.2 DESCRIPCIÓN GENERAL DE LAS INTERFACES GRÁFICAS DE USUARIO. GUI´S

Las interfaces gráficas se desarrollaron centradas en el usuario final, enfocadas en maximizar la usabilidad y la experiencia del usuario. El objetivo final de este diseño de interfaz de usuario es hacer que la interacción entre el usuario y el sistema sea tan simple y eficiente como sea posible, es por ello que en su creación se tomaron en cuenta algunos factores como el logo de la compañía, los colores empresariales y su manejo intuitivo.

La aplicación se basa en cuatro módulos principales mediante los cuales pueden obtener información de acuerdo con su interés, los botones y campos de ingreso de información tienen una estructura de fácil lectura lo cual permite que la interacción con el usuario sea amigable y ágil al momento de usar filtros de búsqueda o generar información

CAPITULO 5. CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO

5.1 CONCLUSIONES

Tras un exhaustivo análisis de las funcionalidades actuales de la aplicación, se concluye que la selección de electrodomésticos representaba un proceso complicado para los usuarios debido a la amplia variedad de categorías que abarcaba, desde electrodomésticos domésticos hasta empresariales y con fines educativos, por lo tanto, se rediseñó la interface gráfica para la selección de los mismos.

Las mejoras significativas implementadas tanto en la aplicación móvil como en el backend permiten mejorar la interacción de los usuarios con la plataforma.

La introducción de una nueva funcionalidad de filtrado por categorías para los electrodomésticos proporciona a los usuarios una mayor eficiencia al realizar sus talleres, agilizando el proceso.

La adopción de la tecnología PWA para el desarrollo de la aplicación ha permitido su ejecución en diversos dispositivos, independientemente del sistema operativo, ampliando así la accesibilidad y alcance a un público más extenso que antes resultaba inalcanzable.

El desarrollo de una aplicación móvil mediante PWA no solo supone la ventaja de un único desarrollo para compatibilidad con IOS y Android, sino también una reducción de costos, ya que no es necesario realizar pagos asociados a la publicación en las tiendas de aplicaciones móviles.

Las aplicaciones móviles basadas en PWA contribuyen a la retención de usuarios al eliminar la necesidad de actualizaciones periódicas, proporcionando un mayor confort a los usuarios, uno de los motivos comunes para el abandono de otras aplicaciones móviles nativas.

La mejora del proceso de despliegue, anteriormente realizado de manera manual, ahora facilita la migración fluida de la aplicación entre proveedores de servicios en la nube, asegurando un funcionamiento sin contratiempos.

5.2 RECOMENDACIONES

En casos de migración hacia otro proveedor de servicios en la nube, distinto al actual Contabo, se recomienda realizar una copia de seguridad exhaustiva de la base de datos y asegurarse de que el nuevo servidor cuente con al menos 6 GB de RAM y 4 CPU Cores para garantizar un rendimiento óptimo.

Para futuras modificaciones en la aplicación, se aconseja mantener el mismo estilo de codificación establecido desde el inicio. Esta práctica contribuirá a preservar la coherencia del código y reducirá los esfuerzos necesarios para la mantenibilidad a largo plazo de la aplicación.

En caso de que la aplicación experimente un aumento significativo en el tráfico de usuarios y, como consecuencia, presente lentitud, se sugiere considerar la migración hacia una arquitectura basada en microservicios. Esta medida permitirá escalar la aplicación de manera más orgánica y eficiente para satisfacer las demandas crecientes.

5.3. TRABAJO FUTURO

Sería sumamente valioso considerar la incorporación de una funcionalidad que genere automáticamente los certificados para los participantes de los diversos talleres. Esto no solo fomentaría la participación, sino que también resaltaría la calidad profesional de las charlas impartidas.

Asimismo, propongo la integración de una nueva función en el sistema, la cual posibilite la modificación, según sea necesario, del factor de emisión. Este parámetro está sujeto a regulaciones periódicas por parte del estado colombiano, por lo que contar con esta capacidad de ajuste sería fundamental para asegurar la conformidad continua con las normativas vigentes, así como para asegurar el correcto funcionamiento y la precisión del sistema.

REFERENCIAS CONSULTADAS

Alcásiga, F., Elizabeth, P., Zapata, P., & Pamela, D. (2022). *"ANÁLISIS COMPARATIVO DE HERRAMIENTAS DE INTEGRACIÓN*.

Arcangeli, J. P., Boujbel, R., & Leriche, S. (2015). Automatic deployment of distributed software systems: Definitions and state of the art. *Journal of Systems and Software*, *103*, 198–218. https://doi.org/10.1016/J.JSS.2015.01.040

Aznar, J. C., Prieto, M., David, V., Besora, B., Panadero, J., Montse, M., & Vizern, S. (2023). *Entornos de desarrollo en la nube para alumnado*.

GitHub Brings Out GitHub Codespaces, GitHub Discussions, GitHub Private Instances And More. (2020, May 8). *Open Source for You*. https://login.bdigital.sena.edu.co/login?url=https://www.proquest.com/magazines/github-brings-out-codespaces-discussions-private/docview/2405844092/se-2?accountid=31491

Muñoz, D. A., Ordóñez, H., & Bucheli, V. (2020). Guideline to implement the CALMS model (DevOps) at mipymes in the software development organizations of the South of Colombia. *Revista Guillermo de Ockham*, *18*(1), 81–91. https://doi.org/10.21500/22563202.4270

Zambrano, F., Patiño, N., & Pino-Correa, F.-J. (2021). Apoyando el despliegue de procesos en el contexto de las pequeñas organizaciones software. *Revista Científica*, *43*(1), 50–63. https://doi.org/10.14483/23448350.18351