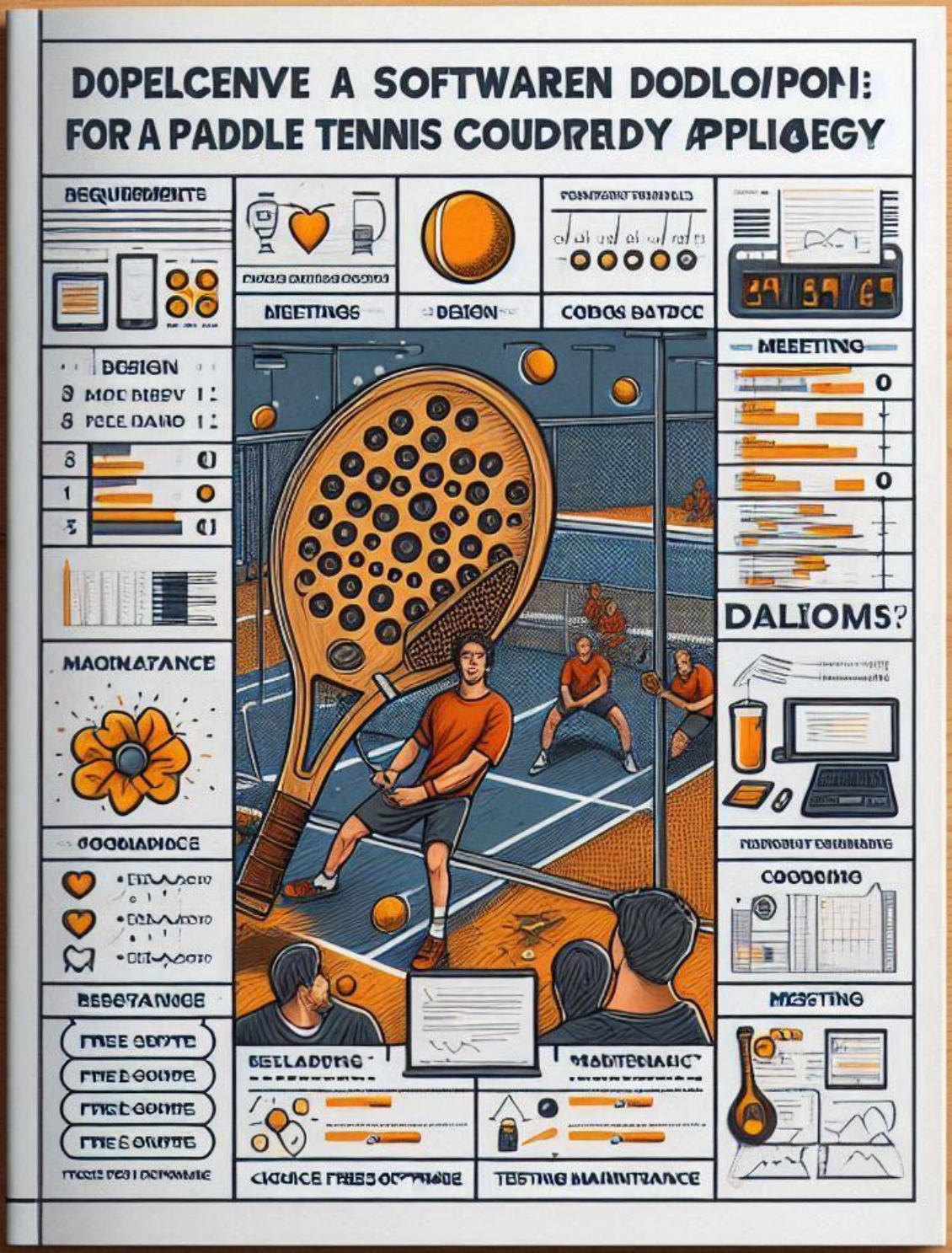


ALUMNO CESUR

Alejandro Muñoz de la Sierra

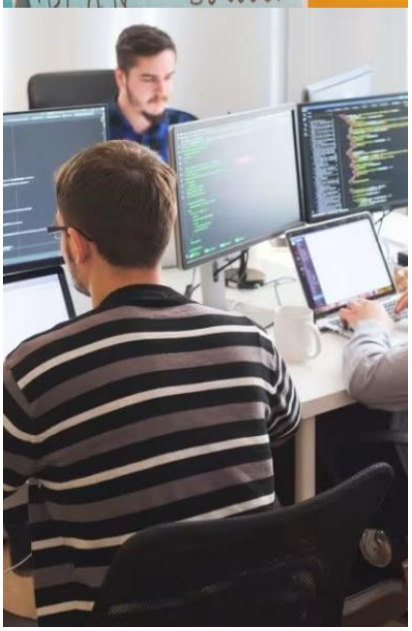
ENTORNOS DE DESARROLLO 2024/25

CASO PRACTICO 1

CONTENIDO

I N T R O D U C C I O N

En esta práctica, nos encargamos del desarrollo de software para una importante cadena de gimnasios con más de 100 centros en toda España. Con el auge del pádel, nuestro cliente necesita una solución digital que facilite la gestión de reservas de pistas. El objetivo principal es crear una aplicación que permita a los usuarios consultar la disponibilidad de las pistas, hacer reservas con o sin luz y procesar pagos con tarjeta, todo desde sus dispositivos móviles, tablets o PCs.



Para llevar a cabo el proyecto, seguiremos la metodología ágil Scrum, organizando el trabajo en sprints que nos permitirán entregar versiones funcionales del software de manera iterativa. Este método no solo asegura entregas regulares, sino que también nos facilita adaptar los requerimientos a medida que surgen nuevas necesidades. Además, emplearemos tecnologías de software libre para garantizar una implementación sencilla, alineada con el compromiso del cliente hacia el uso de herramientas abiertas. El desarrollo abarcará tanto aplicaciones móviles para Android e iOS como una versión accesible desde ordenadores, utilizando tecnologías modernas que aseguren un producto final sólido y escalable.

0 1

ANALISIS Y ESTUDIO DE REQUERIMIENTOS

Es fundamental identificar los requisitos funcionales de la aplicación, como las funciones específicas relacionadas con la reserva de pistas, pagos, y el control de iluminación. Sin embargo, cuando se menciona la identificación completa de los requisitos y la presentación de un plan detallado de análisis, se refiere a incluir tanto los requerimientos funcionales como otros tipos de requisitos que podrían influir en el desarrollo, implementación y mantenimiento de la solución.
Para garantizar que no falte nada relevante, vamos a organizarlos en diversas categorías:

1. Requerimientos Funcionales
Estos son los aspectos clave que mencionaste:

- **Consulta de estado de pistas**: La aplicación debe mostrar si las pistas están disponibles o ya reservadas.
- **Reservas de pistas**: Los usuarios deben poder reservar una pista, con la opción de requerir iluminación si es necesario.
- **Pagos con tarjeta**: Integración de un sistema de pagos en línea mediante tarjetas de crédito.
- **Listado de reservas**: Debe generar informes por hora y por cliente de las reservas realizadas.

Análisis: Aquí nos enfocamos en asegurar que las funciones clave solicitadas por el cliente estén claramente definidas, y en cómo se planificarán cada caso de uso (consulta, reserva, pago, generación de informes) con el máximo detalle.

2. Requerimientos No Funcionales
Se refieren a características esenciales para el funcionamiento general del sistema, aunque no directamente relacionadas con las funciones clave:

- **Compatibilidad con software libre**: Es crucial que las herramientas y tecnologías seleccionadas (frameworks, lenguajes, bases de datos) sean de código abierto.
- **Escalabilidad**: La aplicación debe poder manejar un incremento en el número de usuarios y reservas sin comprometer el rendimiento.
- **Seguridad**: Los datos de los usuarios, especialmente los relacionados con los pagos, deben estar protegidos de acuerdo con las normativas de protección de datos.
- **Disponibilidad**: El sistema debe estar disponible las 24 horas del día para que los usuarios puedan hacer reservas en cualquier momento.
- **Usabilidad**: La interfaz debe ser fácil de usar y accesible, proporcionando una experiencia fluida para los usuarios.

Análisis: Es importante definir métricas de rendimiento (como tiempos de respuesta y carga) y establecer los requisitos de seguridad (como cifrado de datos y cumplimiento de normas de pago).

3. Requerimientos Técnicos

- **Entorno de desarrollo**: Dado que se solicita el uso de software libre, las herramientas elegidas deben alinearse con esta premisa. Ejemplos incluyen:
    - Lenguajes como Python (con frameworks como Django o Flask para el backend), o Node.js para la API.
    - Bases de datos de código abierto como PostgreSQL o MySQL.
    - Herramientas como React Native o Flutter para el desarrollo multiplataforma (iOS y Android).
    - Uso de Git para control de versiones y Docker para la gestión de entornos.
- **Integración con pasarelas de pago**: Se debe seleccionar un servicio de pagos compatible con software libre o que permita una integración sencilla con las tecnologías elegidas (como Stripe o PayPal).

Análisis: Es vital elegir el stack tecnológico de manera que todos los componentes sean compatibles entre sí y con el principio de software libre.

4. Requerimientos de Negocio

- **Estimación de usuarios**: Tener una previsión sobre el número de usuarios ayudará a definir la capacidad del sistema.
- **Presupuesto**: Es importante considerar si existen limitaciones presupuestarias que puedan influir en la elección de tecnologías o la escalabilidad futura del sistema.
- **Mantenimiento**: Es necesario planificar cómo se llevará a cabo el mantenimiento de la aplicación, quién será el responsable y si el cliente tiene la capacidad para gestionarlo o si necesitará soporte externo.
- **Equipo de trabajo**: Evaluar cuántos desarrolladores y especialistas (en seguridad, bases de datos, etc.) serán necesarios para completar el proyecto en tiempo y forma.

Análisis: Aquí se deben considerar factores como el tamaño del equipo, el tiempo disponible para el desarrollo y si se tienen planes de expansión o actualizaciones frecuentes.

5. Requerimientos Legales

- **Protección de datos**: La aplicación debe cumplir con normativas como el GDPR, ya que se gestionarán datos sensibles de usuarios, reservas y pagos.
- **Licencias de software**: Asegurarse de que todas las herramientas empleadas estén bajo licencias apropiadas (GPL, MIT, etc.) para evitar problemas legales con el uso de software libre.

Análisis: Es crucial definir cómo se cumplirá con las normativas legales y asegurarse de que no se vulneren derechos al manejar información personal o utilizar ciertas tecnologías.

Plan Detallado para el Análisis de Requerimientos

Para obtener el máximo rendimiento en este apartado, se debe presentar un plan detallado que cubra cada tipo de requerimiento.

1. **Reuniones con el cliente**:
    - Mantener reuniones iniciales para confirmar y validar todos los requerimientos funcionales y no funcionales.
    - Definir prioridades y asegurarse de que el cliente comprende los compromisos que conlleva el uso de software libre.

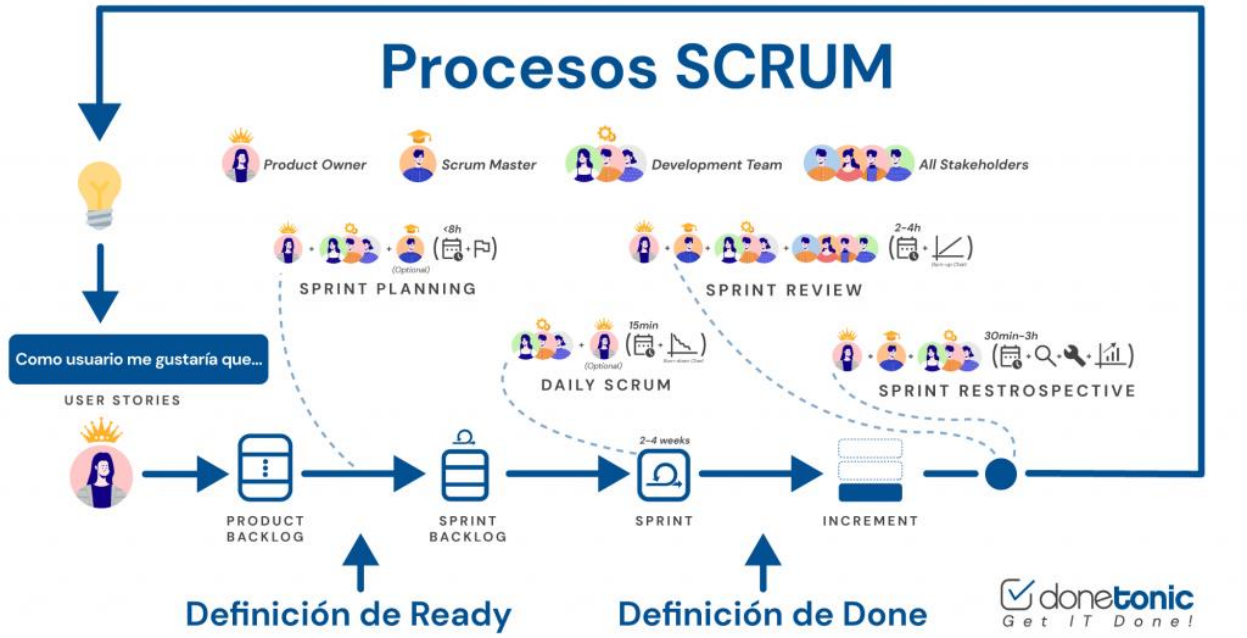
2. **Documentación de requerimientos**:
    - Redactar un documento de especificaciones que incluya todos los puntos: funcionales, no funcionales, técnicos, de negocio y legales.
    - Validar el documento con el cliente para asegurar que todos los requisitos están claros y completos.

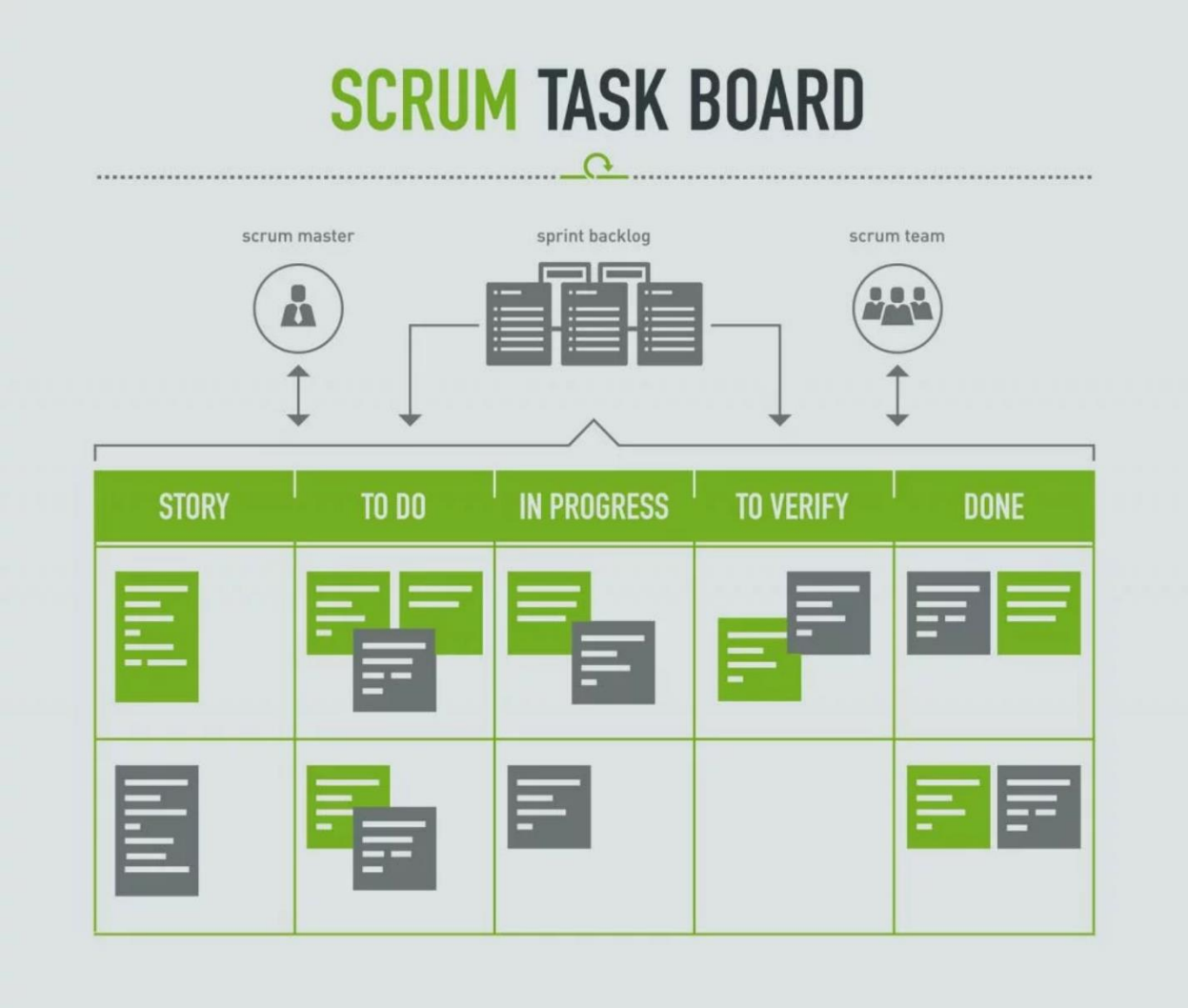
3. **Prototipo inicial**:
    - Desarrollar un prototipo básico (puede ser solo la interfaz) para que el cliente visualice el proyecto y valide los flujos clave.
    - Recopilar feedback y realizar ajustes en los requerimientos antes de avanzar con el desarrollo completo.

4. **Revisión continua**:
    - Establecer revisiones periódicas (al final de cada sprint si se sigue Scrum) para asegurar que los requerimientos se están cumpliendo y verificar si han cambiado las prioridades del cliente.

Conclusión:

El análisis y la identificación de los requerimientos no solo se limitan a los funcionales, sino que también abarcan aspectos técnicos, de negocio, legales y no funcionales. Presentar un plan claro y detallado para abordar cada uno de estos puntos garantizará que no haya sorpresas durante el desarrollo y que el proyecto siga alineado con las expectativas del cliente en todo momento.


Procesos SCRUM
Product Owner
Scrum Master
Development Team
All Stakeholders
SPRINT PLANNING
SPRINT REVIEW
DAILY SCRUM
SPRINT RESTROSPECTIVE
Como usuario me gustaría que...
USER STORIES
PRODUCT BACKLOG
SPRINT BACKLOG
SPRINT
INCREMENT
Definición de Ready
Definición de Done
donetonic
Get IT Done!


SCRUM TASK BOARD
scrum master
sprint backlog
scrum team
STORY
TO DO
IN PROGRESS
TO VERIFY
DONE

02

SELECCION DEL LENGUAJE DE PROGRAMACION Y HERRAMIENTAS

Para cumplir con los requerimientos del proyecto y el compromiso de la empresa con el software libre, se han seleccionado cuidadosamente un lenguaje de programación y herramientas. A continuación se justifica la elección de cada uno:

---

1. Lenguaje de Programación: JavaScript con React para el Frontend y Node.js para el Backend

- **Justificación**:
    - **Software libre**: Tanto React como Node.js son tecnologías de código abierto, lo que cumple con el compromiso de la empresa hacia el uso de software libre.
    - **Popularidad y soporte**: JavaScript es uno de los lenguajes más populares y ampliamente utilizados en el desarrollo web y móvil. Cuenta con una gran comunidad y un ecosistema extenso que proporciona bibliotecas, recursos y soporte.
    - **Desarrollo full-stack**: Usar JavaScript tanto en el frontend como en el backend proporciona uniformidad en el código y una curva de aprendizaje más baja, lo que mejora la eficiencia del equipo de desarrollo.
    - **Desarrollo de aplicaciones móviles**: React Native permite crear aplicaciones nativas para Android e iOS desde un solo código base, lo cual es óptimo considerando que la mayoría de los usuarios utilizarán dispositivos móviles para hacer las reservas.

---

2. Frameworks Seleccionados

Frontend para la Gestión (App Web): React

- **Justificación**:
  - **Versatilidad**: React es uno de los frameworks más populares para construir interfaces de usuario dinámicas y escalables. Ideal para la aplicación de gestión, que requerirá un entorno web eficiente y rápido.
  - **Componentes reutilizables**: React permite crear componentes reutilizables, lo que agiliza el desarrollo y facilita el mantenimiento a largo plazo. Esto es clave para aplicaciones que pueden crecer en funcionalidad.
  - **Bibliotecas y ecosistema**: React tiene un ecosistema enorme con muchas bibliotecas y herramientas que pueden integrarse fácilmente para optimizar el desarrollo.

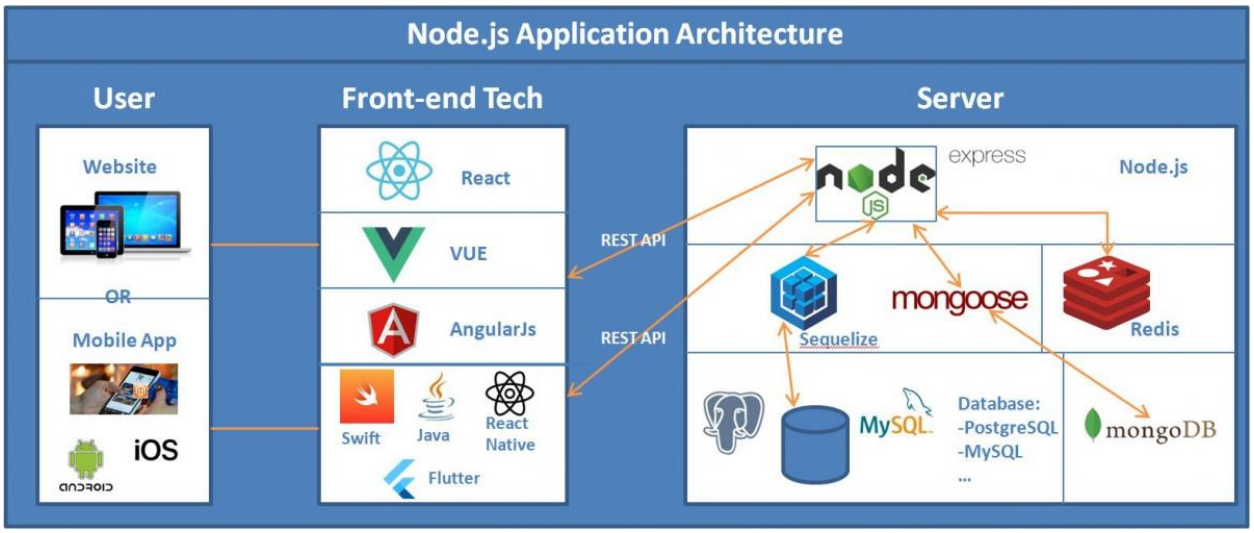
Frontend para Apps Móviles (Android e iOS): React Native

- **Justificación**:
  - **Código compartido**: React Native permite compartir gran parte del código entre las versiones de Android e iOS, lo que ahorra tiempo y esfuerzo en el desarrollo de aplicaciones móviles.
  - **Rendimiento nativo**: A diferencia de soluciones híbridas como Ionic, React Native genera aplicaciones con rendimiento cercano a las aplicaciones nativas, proporcionando una mejor experiencia de usuario.
  - **Soporte extenso**: El ecosistema de React Native cuenta con numerosas bibliotecas y herramientas compatibles para integrar funcionalidades avanzadas, como pagos móviles y notificaciones push.

Backend: Node.js con Express

- **Justificación**:
  - **Alta concurrencia**: Node.js permite manejar múltiples solicitudes de forma eficiente, lo que es esencial en aplicaciones como la reserva de pistas que podrían recibir muchas conexiones simultáneamente.

- **Express.js**: Un framework minimalista para Node.js que facilita la creación de APIs RESTful, consumibles tanto por la app de gestión web (React) como por las aplicaciones móviles (React Native).
- **Escalabilidad**: Node.js facilita la escalabilidad, adaptándose a las necesidades crecientes de la empresa a medida que el número de usuarios aumenta.



3. Herramientas de Desarrollo

Entorno de Desarrollo: VSCode (Visual Studio Code)

- **Justificación**:
  - **Popularidad y extensiones**: VSCode es uno de los editores más populares para desarrollo de software libre y ofrece una amplia variedad de extensiones para JavaScript, React y React Native, facilitando la depuración y el desarrollo en tiempo real.
  - **Integración con Docker**: VSCode permite integrar herramientas como Docker, facilitando la creación de entornos de desarrollo consistentes para la aplicación.
  - **Soporte para Git**: VSCode cuenta con integración nativa con Git, lo que facilita la gestión de versiones del código durante los diferentes sprints.

---

4. Integración con Sistemas de Pago: RedSys para Pagos en España

- **Justificación**:
    - **Preferencia local**: Dado que la empresa opera en España, se optará por la integración con RedSys para procesar pagos, ya que es la pasarela de pagos más utilizada en el país y permite realizar pagos con tarjeta de forma segura.
    - **Bajas comisiones**: Comparado con plataformas globales como Stripe, RedSys ofrece comisiones más competitivas para transacciones locales en España.
    -

---

5. Bases de datos: PostgreSQL

- **Motivación**: PostgreSQL es una base de datos relacional muy valorada en el mundo del código abierto. Su capacidad para manejar proyectos de diferentes escalas, sumado a su compatibilidad con estándares y su flexibilidad en la gestión de esquemas y consultas avanzadas, la convierten en una opción ideal para este proyecto.
- **Ventajas**:
    - Escalabilidad: Esta base de datos soporta grandes volúmenes de información, lo que resulta beneficioso a medida que crece el número de usuarios y reservas.
    - Compatibilidad con Node.js: Existen bibliotecas y módulos, como pg-promise, que facilitan la integración entre el backend y PostgreSQL.
    - Consultas avanzadas: Su capacidad para manejar consultas complejas es especialmente útil cuando se requieren informes detallados, como el historial de reservas por hora o por cliente.

---

6. Herramientas para despliegue y contenedores: Docker

- **Motivación**: Docker se ha posicionado como una de las plataformas clave para el desarrollo y despliegue de aplicaciones, especialmente en el mundo del software libre. Permite empaquetar aplicaciones con todas sus dependencias, lo que facilita la creación de entornos consistentes entre desarrollo y producción.

- **Ventajas**:
    - Implementación ágil: Docker simplifica el despliegue rápido de aplicaciones, permitiendo que los componentes como el frontend, backend y la base de datos se instalen sin complicaciones.
    - Escalabilidad: Facilita la ampliación de servicios, como las bases de datos o APIs, cuando sea necesario.

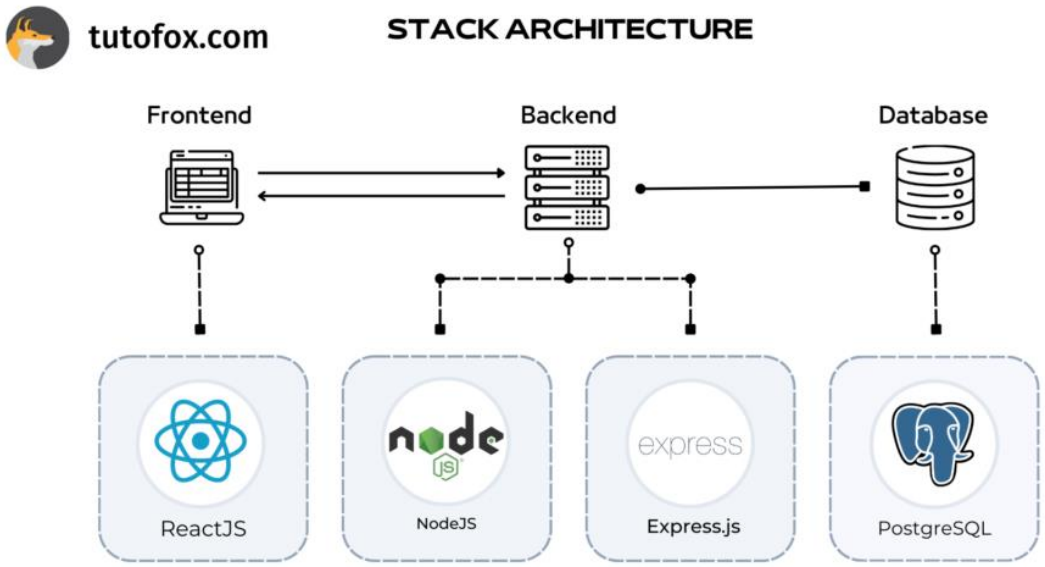
---

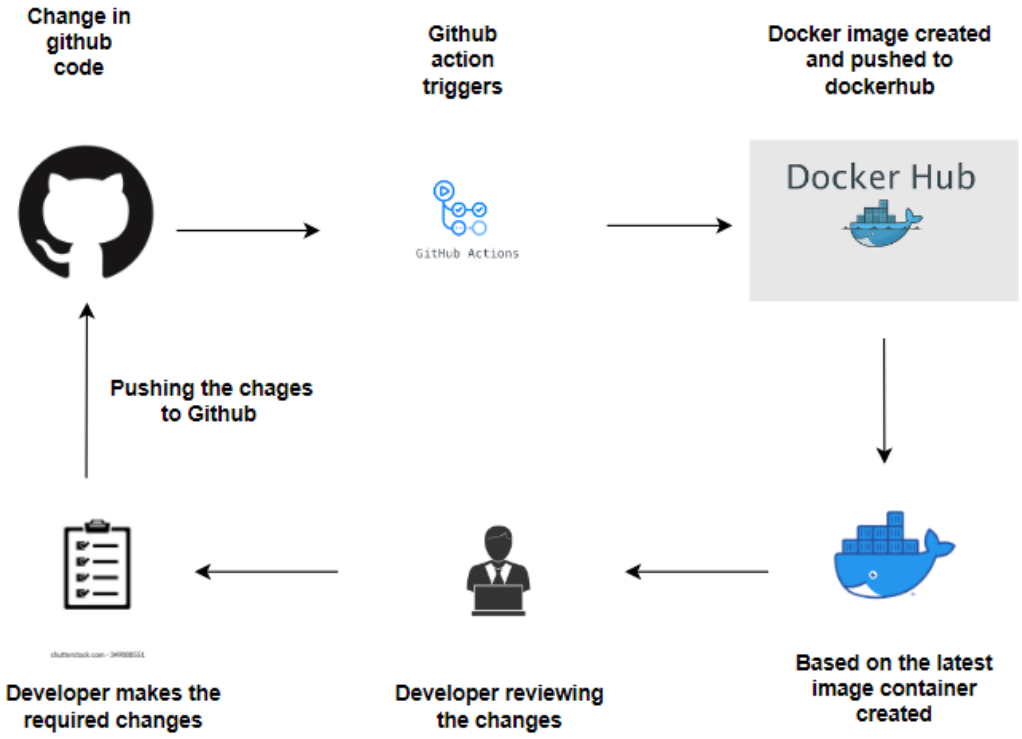
7. Sistema de control de versiones: Git

- **Motivación**: Git se ha convertido en el estándar para la gestión de versiones, permitiendo a los equipos de desarrollo colaborar eficientemente. Su naturaleza de código abierto lo hace una excelente elección para proyectos basados en software libre.
    - **Repositorios alojados en GitHub o GitLab:** Ambos servicios permiten almacenar el código de forma gratuita y ofrecen herramientas adicionales para gestionar proyectos ágiles.

Este enfoque asegura que el proyecto se alinee con las necesidades de la empresa y el compromiso con el uso de tecnologías de código abierto.

Conclusión: Las herramientas y lenguajes seleccionados no solo cumplen con los requisitos de software libre de la empresa, sino que también aseguran escalabilidad, rendimiento y facilidad de desarrollo tanto en la web como en las aplicaciones móviles, maximizando la eficiencia del equipo y garantizando una experiencia de usuario óptima.


tutofox.com
STACK ARCHITECTURE
Frontend
Backend
Database
ReactJS
NodeJS
Express.js
PostgreSQL


Change in github code
Github action triggers
Docker image created and pushed to dockerhub
GitHub Actions
Docker Hub
Pushing the chages to Github
Developer makes the required changes
Developer reviewing the changes
Based on the latest image container created

0 3

F A S E D E D I S E Ñ O

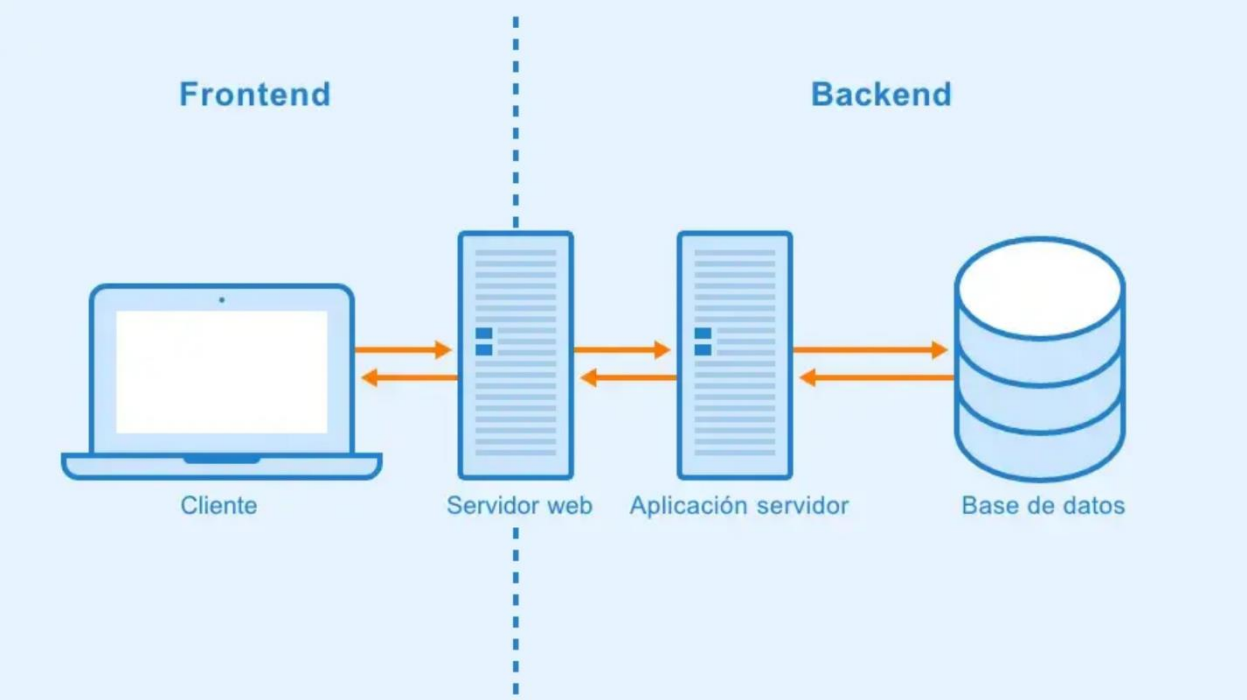
El objetivo en esta fase es establecer la arquitectura general del sistema, que incluye la interfaz de usuario, la lógica del backend, la base de datos y el método de despliegue. Durante esta etapa, se planificará la estructura de la UI, las funcionalidades clave del backend, y el sistema de pagos, eligiendo las tecnologías más adecuadas para garantizar un desarrollo eficaz y escalable. Se emplearán herramientas como React, React Native, Node.js y Docker, alineándose con los principios de software libre y optimización del rendimiento.

- **Actividades**:
    - **Diseño del frontend (interfaz de usuario)**:
        - **Tecnologías y herramientas**: React será utilizado para desarrollar la interfaz web de la gestión de pistas, mientras que React Native se usará para las aplicaciones móviles (disponibles en iOS y Android).
        - **Estructura modular**: Implementación de componentes reutilizables para mejorar la eficiencia del desarrollo y el mantenimiento.
    - **Diseño del backend (lógica del negocio)**:
        - **Tecnologías y herramientas**: Node.js con el framework Express se empleará para construir el servidor y gestionar las API.
        - **Arquitectura RESTful**: Se asegura una comunicación eficiente entre las aplicaciones de frontend (React y React Native) y el backend.
    - **Diseño de la base de datos**:
        - **Tecnología**: Se utilizará PostgreSQL como base de datos.
        - **Estructura**: Diseño de tablas para gestionar las reservas, usuarios, pistas, y procesar los pagos.

- **Sistema de despliegue**:
  - **Herramienta**: Docker se usará para empaquetar y desplegar la aplicación en entornos de desarrollo, pruebas y producción.

- **Pasarela de pagos**:
  - **Herramienta**: Se optará por RedSys para el procesamiento de pagos, dado que es más adecuado para el mercado español en comparación con Stripe, garantizando transacciones seguras.

Este diseño detallado asegurará que todos los componentes del sistema estén bien integrados y optimizados para el crecimiento y la escalabilidad del proyecto.



0 4

F A S E D E I M P L E M E N T A C I O N Y C O D I F I C A C I O N

El objetivo de esta fase es desarrollar el software conforme al diseño previamente definido, aplicando las mejores prácticas de desarrollo, como los principios SOLID. Se abordarán tanto el desarrollo del frontend y las aplicaciones móviles como la creación del backend, la integración de la base de datos y la pasarela de pagos. También se garantizará un entorno eficiente y escalable utilizando Docker para el despliegue.

- **Actividades**:
    - **Implementación del frontend:**
        - Desarrollo de la interfaz de usuario: Construcción de la UI para la gestión de las pistas utilizando React, asegurando que la aplicación web sea dinámica y fácil de usar.
        - Desarrollo de las aplicaciones móviles: Creación de las aplicaciones móviles para Android e iOS utilizando React Native, permitiendo una experiencia fluida para los usuarios que realicen reservas desde dispositivos móviles.

    - **Implementación del backend:**
        - **Desarrollo de las APIs RESTful:** Uso de Node.js con el framework Express para manejar las operaciones principales del sistema, como la consulta de pistas, la gestión de usuarios y el procesamiento de pagos.

    - **Integración con la base de datos:**
        - **Gestión de datos**: Implementación de consultas y operaciones en PostgreSQL para manejar la información de reservas, usuarios, pistas, y pagos de manera eficiente y segura.

    - **Integración con la pasarela de pagos:**

- Proceso de pagos: Incorporación de RedSys para permitir el procesamiento de pagos con tarjeta, asegurando que las transacciones sean seguras y cumplan con las normativas locales en España.

- **Despliegue de la aplicación:**
  - **Uso de Docker:** Emplear Docker para desplegar tanto el backend como el frontend, asegurando que la aplicación sea escalable y que el entorno de desarrollo sea coherente con el entorno de producción.

De este modo, se seguirá el plan de desarrollo establecido, garantizando que todos los componentes del sistema estén bien integrados y optimizados para ofrecer una experiencia de usuario de alto rendimiento.

0 5

F A S E D E P R U E B A S

El objetivo de esta fase es asegurar que todas las funcionalidades del software cumplen con los requerimientos establecidos antes de su entrega final. Esta fase se enfoca en validar el correcto funcionamiento de los componentes, la integración de los módulos, y el rendimiento y seguridad del sistema.

- **Actividades**:
    - **Pruebas unitarias**:
        - **Evaluación de componentes aislados**: Se realizarán pruebas individuales en cada componente del sistema, tanto en el frontend como en el backend, para verificar que funcionen correctamente de manera independiente.

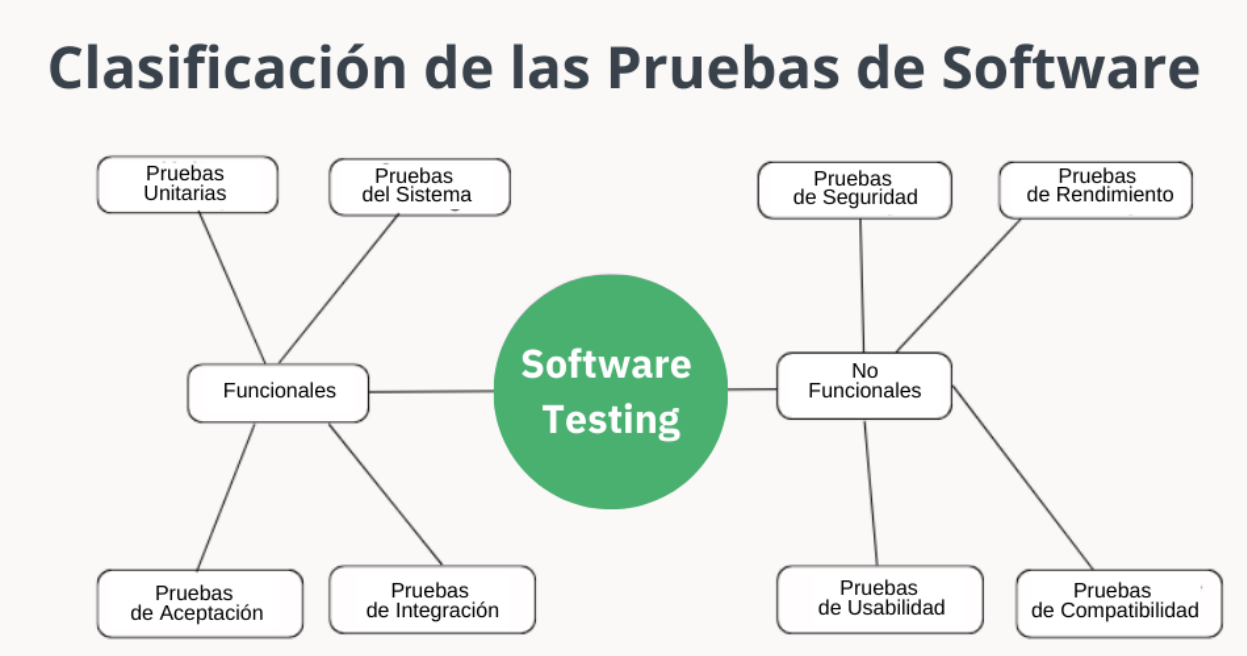
    - **Pruebas de integración**:
        - **Validación de la interacción entre módulos**: Se comprobará que los diferentes componentes del sistema (frontend, backend, base de datos, y pasarela de pagos) se comunican e integran correctamente entre sí.

    - **Pruebas de sistema**:
        - **Simulación de condiciones reales**: Se ejecutarán pruebas que simulen el uso real del sistema, evaluando el comportamiento general de la aplicación bajo condiciones de carga similares a las del entorno de producción, como múltiples usuarios reservando pistas o realizando pagos al mismo tiempo.

    - **Pruebas de rendimiento y seguridad**:
        - **Eficiencia y seguridad**: Se evaluará la capacidad de la aplicación para manejar múltiples usuarios simultáneamente sin degradación del rendimiento. Además, se garantizará la seguridad en la transacción de pagos, asegurándose de que la

integración con RedSys cumpla con los estándares de protección de datos y transacciones seguras.

Con esta fase, se verificará que el software está listo para ser utilizado por los usuarios finales, sin errores y cumpliendo con las expectativas de rendimiento y seguridad necesarias.



0 6

F A S E D E D E S P L I E G U E

El objetivo de esta fase es poner en funcionamiento el software para su uso en un entorno de producción, asegurando que tanto el backend como el frontend, la base de datos, y las aplicaciones móviles estén operativos y accesibles para los usuarios finales.

- **Actividades**:
    - **Despliegue del backend**:
        - **Uso de Docker**: Se procederá al despliegue del backend, desarrollado en Node.js con Express, utilizando Docker para garantizar que el entorno de producción sea consistente con el de desarrollo y facilitar la escalabilidad de los servicios.

    - **Despliegue del frontend**:
        - **Hosting de la aplicación web**: El frontend, desarrollado en React para la gestión de reservas, será desplegado en un servidor o plataforma de hosting adecuada. Se garantizará que la aplicación esté accesible y optimizada para su uso a través de navegadores web.

    - **Configuración de la base de datos**:
        - **PostgreSQL**: Se configurará la base de datos PostgreSQL en el servidor, asegurando que esté lista para gestionar las operaciones de consulta y almacenamiento de datos, como las reservas de pistas y la información de los usuarios.

    - **Documentación técnica**:
        - **Instrucciones detalladas**: Se generará la documentación técnica del proyecto, que incluirá las instrucciones paso a paso para el despliegue del software, la configuración de Docker, el manejo de la base de datos, y los procedimientos de mantenimiento y actualización del sistema.

- **Publicación de las aplicaciones móviles**:
  - **App Store y Google Play**: Las aplicaciones móviles, desarrolladas en React Native para iOS y Android, serán empaquetadas y publicadas en las tiendas de aplicaciones correspondientes (App Store y Google Play), asegurando que estén accesibles para su descarga y uso por parte de los usuarios finales.

Esta fase garantiza que el sistema esté completamente operativo y disponible tanto en web como en dispositivos móviles, listo para ser utilizado por los usuarios finales.
4o

---

0 7

F A S E D E M A N T E N I M I E N T O

El objetivo de esta fase es asegurar que la aplicación continúe funcionando de manera óptima tras su despliegue, manteniéndola actualizada, segura y libre de errores, adaptándose a las necesidades cambiantes de la empresa y los usuarios.

- **Actividades**:
  - **Corrección de errores**:
    - **Solución de problemas post-despliegue**: Se identificarán y corregirán los errores o problemas que puedan surgir tras la puesta en producción, garantizando que la aplicación mantenga un rendimiento estable y confiable.

  - **Actualizaciones y mejoras**:
    - **Adaptación a nuevas necesidades**: Se realizarán actualizaciones periódicas y se implementarán nuevas funcionalidades o mejoras de acuerdo con los requerimientos

de la empresa y el feedback de los usuarios. Esto puede incluir la optimización de la interfaz de usuario, nuevas características para la gestión de pistas, o mejoras en el rendimiento de la aplicación.

- **Mantenimiento de la seguridad**:
  - **Protección de datos y pagos**: Se realizarán auditorías de seguridad periódicas para asegurar que los datos de los usuarios y la información de los pagos se encuentren protegidos de acuerdo con las normativas y estándares vigentes. Además, se actualizarán los sistemas de seguridad para prevenir vulnerabilidades y garantizar transacciones seguras mediante la pasarela de pagos RedSys.

Esta fase asegura la continuidad operativa del sistema, su evolución conforme a las necesidades del negocio, y el mantenimiento de altos estándares de seguridad y rendimiento.

0 8

C O N C L U S I O N E S

El desarrollo de la aplicación para la reserva de pistas de pádel, siguiendo la metodología ágil SCRUM, ha proporcionado un enfoque organizado y eficiente para cumplir con los objetivos de la empresa. A través de fases bien estructuradas, hemos identificado los requerimientos clave, seleccionado herramientas y lenguajes de programación adecuados, y planificado cada etapa del ciclo de vida del software. Todo esto ha sido llevado a cabo respetando los principios del software libre, lo que refuerza el compromiso con soluciones abiertas y accesibles.

La elección de React y React Native para el desarrollo tanto del frontend web como de las aplicaciones móviles (iOS y Android), junto con Node.js y Express para el backend, ha permitido implementar una arquitectura moderna, escalable y eficiente. Docker ha facilitado el despliegue y escalabilidad del sistema, mientras que la integración de RedSys para los pagos ha asegurado un entorno de ejecución confiable y seguro, especialmente adaptado al mercado español.

Durante todo el proceso, se ha mantenido un enfoque en la calidad del código y en las mejores prácticas de desarrollo, aplicando los principios SOLID y realizando pruebas exhaustivas para garantizar que el software cumpla con los requisitos funcionales y no funcionales. Además, se han establecido procedimientos claros para el mantenimiento del sistema, lo que garantiza que la aplicación continúe operando de manera óptima y pueda evolucionar de acuerdo con las necesidades futuras del cliente o los avances tecnológicos.

En resumen, este proyecto ha logrado diseñar, desarrollar y desplegar una solución integral y robusta, que facilita la gestión eficiente de las reservas de pistas de pádel. Esto asegura tanto la satisfacción del cliente como una experiencia positiva para los usuarios finales de la aplicación, sentando las bases para futuras mejoras y adaptaciones del sistema.

0 9

R E F E R E N C I A S

https://asana.com/es/resources/what-is-scrum

https://ilimit.com/blog/metodologia-scrum/

https://www.youtube.com/watch?v=sLexw-z13Fo&t=127s

https://rootstack.com/es/blog/nodejs-vs-react-cual-necesito-para-crear-un-sitio-web#:~:text=js%20est%C3%A1%20dise%C3%B1ado%20para%20crear,de%20piezas%20individuales%20llamadas%20componentes.

https://www.freecodecamp.org/espanol/news/react-js-vs-react-native-cual-es-la-diferencia/#:~:text=React%20JS%20se%20utiliza%20para,es%20decir%2C%20aplicaciones%20m%C3%B3viles%20multiplataformas)

https://keepcoding.io/blog/ventajas-de-usar-node-js/#:~:text=Gracias%20a%20su%20arquitectura%20basada,tiempo%20real%20y%20sin%20problemas.

https://www.youtube.com/watch?v=lWQ69WX7-hA

https://www.youtube.com/watch?v=_xL4oil0gQo

https://blog.infranetworking.com/5-razones-por-las-cuales-debes-usar-postgresql/

https://blog.hubspot.es/website/framework-desarrollo-web

https://tekla.io/blog/que-es-redsys/