UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES E INFORMÁTICOS

Tema:

SITIO WEB PARA LA GESTIÓN DEL RECLUTAMIENTO Y SELECCIÓN DE FUTBOLISTAS EN LA CIUDAD DE AMBATO

Trabajo de Titulación. Modalidad: Proyecto de Investigación, presentado previo a la obtención del título de Ingeniero en Sistemas Computacionales e Informáticos

ÁREA: Software
LÍNEA DE INVESTIGACIÓN: Desarrollo de software
AUTOR: Génesis Noemi Rubira Ramírez
TUTOR: Ing. Edwin Hernando Buenaño Valencia, Mg

Ambato - Ecuador
marzo - 2023

APROBACIÓN DEL TUTOR

En calidad de Tutor del Trabajo de Titulación con el Tema: SITIO WEB PARA LA GESTIÓN DEL RECLUTAMIENTO Y SELECCIÓN DE FUTBOLISTAS EN LA CIUDAD DE AMBATO, desarrollado bajo la modalidad Proyecto de Investigación por la señorita Génesis Noemi Rubira Ramírez, estudiante de la Carrera de Ingeniería en Sistemas Computacionales e Informáticos, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que la estudiante ha sido tutorada durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 15 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y el numeral 7.4 del respectivo instructivo.

Ambato, marzo 2023

Ing. Edwin Hernando Buenaño Valencia, Mg.

TUTOR

AUTORÍA

El presente Proyecto de Investigación titulado: SITIO WEB PARA LA GESTIÓN DEL RECLUTAMIENTO Y SELECCIÓN DE FUTBOLISTAS EN LA CIUDAD DE AMBATO es absolutamente original, auténtico y personal. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, marzo 2023

Génesis Noemí Rubira Ramírez

CC: 1805714167

AUTOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este Trabajo de Titulación como un documento disponible para la lectura, consulta y procesos de investigación.

Cedo los derechos de mi Trabajo de Titulación, con fines de difusión pública. Autorizo además su reproducción dentro de las regulaciones de la Universidad Técnica de Ambato.

Ambato, marzo 2023

Génesis Noemí Rubira Ramírez

CC: 1805714167

AUTOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del Informe Final del Trabajo de Titulación presentado por la señorita Génesis Noemi Rubira Ramírez, estudiante de la Carrera de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado SITIO WEB PARA LA GESTIÓN DEL RECLUTAMIENTO Y SELECCIÓN DE FUTBOLISTAS EN LA CIUDAD DE AMBATO, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 17 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y al numeral 7.6 del respectivo instructivo. Para cuya constancia suscribimos, conjuntamente con la señora Presidenta del Tribunal.

Ambato, marzo 2023


Firmado electrónicamente por:
ELSA PILAR URRUTIA URRUTIA

Ing. Pilar Urrutia, Mg

PRESIDENTE DEL TRIBUNAL


Firmado electrónicamente por:
DENNIS VINICIO CHICAIZA CASTILLO

Ing. Dennis Chicaiza, Mg
PROFESOR CALIFICADOR


Firmado electrónicamente por:
CARLOS ISRAEL NUNEZ MIRANDA

Ing. Carlos Núñez, Mg
PROFESOR CALIFICADOR

DEDICATORIA

El presente trabajo de investigación lo dedico principalmente a mi madre, Georgina Ramirez y hermanos, Esteban y Andres quienes han sido el pilar fundamental para lograr cada una de mis metas y me han brindado su apoyo incondicional durante todo este proceso.

A cada una de las personas que me alentaron para seguir adelante y no rendirme.

Génesis Noemi Rubira Ramírez

AGRADECIMIENTO

Primeramente, agradezco a Dios por permirtirme cumplir mis objetivos y poner en mi camino a personas que me han servido de guía y apoyo para llevar a cabo el presente trabajo de investigaciòn.

A mi madre, hermanos y seres queridos, por estar siempre a mi lado y brindarme su confianza y apoyo.

A los docentes de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial por impartirme sus conocimientos y consejos a lo largo de mi carrera universitaria.

A mi tutor, el Ing. Buenaño Valencia Edwin Hernando, Mg. por haber sido mi guia durante este proceso.

Génesis Noemi Rubira Ramírez

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN EJECUTIVO

El fútbol ha trascendido desde épocas antiguas y con el paso del tiempo se ha convertido en uno de los deportes más populares en distintos niveles, como son: deportivo, entretenimiento, social y económico; debido a dicha popularidad existe un gran número de personas que se apasionan por ser futbolistas, por lo que uno de sus pilares fundamentales es la búsqueda de nuevos talentos, misma que ha ido evolucionando y mejorando, sin embargo, actualmente este proceso genera ciertas limitaciones en cuanto a la gestión de la información de cada participante y el tiempo que estos tienen para el desarrollo de sus pruebas, limitando de esta forma su desempeño; es por esto que surge la idea del presente proyecto titulado "Sitio web para la gestión del reclutamiento y selección de fultbolistas en la ciudad de Ambato" con la finalidad de optimizar recursos y tiempo tanto de los clubes de fútbol como de los postulantes a la hora de realizar el proceso de selección y reclutamiento por medio de la implementación de un sitio web.

El presente proyecto se desarrolló con la arquitectura cliente-servidor; la parte del cliente (front-end) está desarrollado en Visual Estudio Code con el framework angular js; el servidor (back-end) es un api rest para el cual se han implementado varias tecnologías, empezando principalmente por node.js y express.js para su creación y adicionando librerías, herramientas y dependencias como sequelize para el manejo de la base de datos, multer y ffprobe para la gestión de archivos de video e imagen y node-cron para las tareas programadas que el sistema requiere.

Para administrar y gestionar el presente trabajo de investigación se utilizó la metodología ágil XP (Programación Extrema), la cual está orientada a pequeños y medianos equipos y se basa en valores, principios y prácticas, con la finalidad de producir software de calidad y adaptable a los requisitos cambiantes.

Palabras clave: Deporte, talento deportivo, framework, Angular js, Node.js, Express.js, desarrollo de software.

ABSTRACT

Soccer has transcended since ancient times and with the passage of time it has become one of the most popular sports at different levels, such as: sports, entertainment, social and economic; Due to this popularity, there is a large number of people who are passionate about being soccer players, so one of its fundamental pillars is the search for new talents, which has been evolving and improving, however, currently this process generates certain limitations in regarding the management of the information of each participant and the time they have to develop their tests, thus limiting their performance; This is why the idea of the present project entitled "Website for the management of the recruitment and selection of soccer players in the city of Ambato" arises with the purpose of optimizing resources and time of both soccer clubs and applicants at the time to carry out the selection and recruitment process through the implementation of a website.

This project was developed with the client-server architecture; the client part (front-end) is developed in Visual Studio Code with the angular js framework; the server (back-end) is a rest api for which various technologies have been implemented, starting mainly with node.js and express.js for its creation and adding libraries, tools and dependencies such as sequelize for database management, multer and ffprobe for the management of video and image files and node-cron for the scheduled tasks that the system requires.

To administer and manage this research work, the agile XP (Extreme Programming) methodology was improved, which is aimed at small and medium-sized teams and is based on values, principles and practices, in order to produce quality software that is adaptable to changing requirements.

Keywords: Sport, sports talent, framework, Angular js, Node.js, Express.js, software development.

CAPÍTULO I

MARCO TEÓRICO

1.1. Tema de Investigación

SITIO WEB PARA LA GESTIÓN DEL RECLUTAMIENTO Y SELECCIÓN DE FUTBOLISTAS EN LA CIUDAD DE AMBATO.

1.2. Antecendentes Investigativos

Como antecedentes investigativos se pueden mencionar los siguientes:
Palacios Menéndez Christian Andrés en su tesis de grado cuyo tema es "Diseño e Implementación de Plataforma Web para Control y Seguimiento de Actividades del Personal del Sistema Canal Radio y Televisión de la Universidad Católica Santiago de Guayaquil" (2019) indica que el desarrollo de aplicaciones web tiene un enfoque distinto al del modelo clásico para la creación de software debido a que demandan el uso de características importantes las cuales destacan la funcionalidad del software, por lo cual la utilización de un modelo de desarrollo incremental permite que los desarrollares administren los requerimientos esenciales para liberarlos en un orden planificado de modo que se mantenga un esquema funcional para el diseño y la creación de la aplicación web además concluye que el uso del framework Laravel optimiza la ejecución de procesos al transferir información desde una base de datos a una aplicación final.[1]
De Luca Agustín Marcelo y Vignolo Agustín Octavio en su tesis de grado titulado "Machi: Aplicación móvil para el acercamiento de la tecnología al deporte" elaborada en la Universidad Nacional de la Plata en el año 2021, donde explica sobre el deporte y la tecnología partiendo del estudio de la historia de la tecnología en el deporte detallando los avances notables en deportes y las aplicaciones orientadas a la mejora y desarrollo de las capacidades físicas de los deportistas concluye que esta información evidencia lo indispensable que es el uso de la tecnología para el desarrollo del deporte de más alto nivel al igual que la utilidad de esta en la práctica de los mismos.[2]
Josué Danilo Delgado Navas y Wilson Hernando Bravo Navarro en su artículo titulado "Propuesta de criterios de selección de talentos en la escalada deportiva"

elaborado en el año 2021 indican que la selección de talentos deportivos conlleva una gran responsabilidad ya que involucra a varias personas, partiendo desde los padres, dirigentes y entrenadores donde el fin común es aprovechar las capacidades técnicas y deportivas de los deportistas para desarrollarlas y así alcanzar logros importantes en una determinada disciplina; por otra parte añaden que en la actualidad los caza talentos buscan deportistas que tengan una edad temprana con el objetivo de desarrollar sus cualidades innatas mediante la preparación profesional.[3]

1.2.1. Contextualización del problema

Uno de los pilares fundamentales del fútbol es la búsqueda de nuevos talentos, razón por la cual para la detección de estos talentos se toman en cuenta varios factores como son: físicos, fisiológico, psicológico, cognitivo, social, la genética, edad de maduración, habilidades perceptivas, habilidades tácticas y el tiempo de práctica.

Tradicionalmente la identificación y selección de estos talentos ha estado basada en evaluaciones subjetivas llevadas a cabo por entrenadores experimentados, sin embargo, estas evaluaciones han sido completadas con pruebas objetivas.

Durante las últimas décadas los procesos de identificación y selección de futbolistas de alto rendimiento ha mejorado debido a que los avances de la ciencia y la tecnología reflejaron la necesidad de investigar el fútbol como deporte[4].

En la actualidad el proceso de selección y reclutamiento de futbolistas se centra en el campo físico-táctico donde en muchas ocasiones la cantidad de futbolistas que los clubes deben evaluar es elevada para pocas plazas disponibles dentro de sus planteles lo que imposibilita el almacenamiento, gestión y revisión detallada de la información de cada postulante.

Por otra parte, una gran cantidad de jugadores pierdan la oportunidad de ser seleccionados por su bajo desenvolvimiento en las pruebas de campo debido a factores físicos, emocionales y técnicos ya que muchos de estos jugadores se trasladan desde ciudades lejanas, pasando por noches de desvelo o en lugares de alojamiento poco cómodos y tienen una alimentación inestable[5].

1.2.2. Fundamentación teórica

1.2.2.1. Desarrollo de software

Es un proceso de aprendizaje social debido a que se basa en convertir el conocimiento adquirido del diálogo por medio de la interacción entre usuarios

y diseñadores, usuarios y herramientas, y diseñadores y herramientas a un sistema.[6]

1.2.2.2. Metodologías de desarrollo

Es el marco de trabajo que se sigue para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información, permitiendo gestionar y administrar un proyecto por medio de sus componentes (fases, documentación, técnicas y herramientas, métodos, y control y evaluación); sus objetivos principales son:

- Asegurar la uniformidad y calidad del desarrollo y del sistema.
- Satisfacer las necesidades del usuario.
- Optimizar el nivel de rendimiento y eficiencia del personal de desarrollo.
- Ajustarse tanto a los plazos como a los costes previstos en la planificación.
- Facilitar el mantenimiento.
- Definir actividades a llevar a cabo.
- Proporcionar puntos de control y revisión.
- Documentar el proceso de desarrollo.
- Identificar cambios.[7]

1.2.2.3. Framework

Un framework es un subsistema o conjunto de librerías que brindan funcionalidades estándares a cualquier sistema, además provee una estructura de carpetas y archivos que sirven para la organización del código, arquitectura para el desarrollo de un proyecto, seguridad, robustez, soporte y un conjunto de buenas prácticas de programación.

Angular js

Framework JavaScript diseñado con una arquitectura orientada a módulos y componentes por lo cual sus elementos principales son: módulos, componentes, servicios y directivas, es utilizado en el desarrollo web y aplicaciones móviles híbridas(ionic), implementa el lenguaje de programación TypeScript.

Su instalación se realiza por medio de la línea de comandos mediante el gestionador de paquetes que viene por defecto con Node.js llamado npm.

Angular promueve y usa patrones de diseño de software. En concreto implementa lo que se llama MVC (Model View Controller- Modelo Vista Controlador) poco definido. Almacena el código de interfaz y el de la lógica de negocio por separado.
Características:

- Curva de aprendizaje alta
- Shadow DOM
- Documentación completa, simple y gratuita
- Compilador Bazel

Ventajas:

- Marco E2E completo y con opiniones.
- Fácil de mantener el código.
- El enlace de datos bidireccional evita la intervención del desarrollador.
- Renderizado del lado del servidor (SSR) listo para usar con soporte de caché reduciendo la carga de la CPU.
- El marco está diseñado para ser altamente comprobable.
- El uso de directivas, componentes y elementos ayuda a limpiar organizando el código.
- Angular CLI facilita el seguimiento de las mejores prácticas

Desventajas:

- Tiene tiempos de carga iniciales lentos y todavía lucha con complejos cambia frecuentemente de página. Sin embargo, SSR alivia el problema.
- La organización obstinada del código en realidad hace que sea difícil manejar los alcances de entidades complejas.
- El apoyo de la comunidad está disminuyendo lentamente.

React js
Framework de JavaScript, su arquitectura es basada en componentes y trabaja mediante el Virtual DOM; utilizado para el desarrollo web y aplicaciones móviles nativas con react native, soporta Typescript. Implementa el patrón MVC (Model View Controller- Modelo Vista Controlador).
Características:

- Curva de aprendizaje media
- Gran popularidad y comunidad de desarrolladores
- Rápido renderizado
- Flujo de datos unidireccional
- Componentes con estado
- Virtual DOM

Ventajas:

- Fácil de aprender.
- Sin opiniones y muy flexible. Compatible con Typescript.
- La implementación de Virtual DOM lo convierte en uno de los clientes más rápidos.
- El flujo de datos unidireccional evita que el padre se vea afectado por los hijos.
- Es relativamente más fácil migrar entre versiones.
- Create-react-app simplifica la creación de una nueva aplicación desde cero.

Desventajas:

- La documentación oficial sigue siendo deficiente.
- Dado que hay muchos componentes listos para usar para elegir se convierte en responsabilidad del desarrollador evaluarlos.
- Dado que es un marco de solo vista, requiere un esfuerzo considerable para integrarse con otras capas de MVC[8].

Vue.js

Framework JavaScript progresivo, permite desarrollar un proyecto por módulos y no tener dependencias debido a que las librerías están desacopladas. Utiliza el patrón Model-View-ViewModel (MVVM) o conocido también con el nombre Model-View-Whatever, se divide en tres partes: modelo, vista y ViewModel o modelo de la vista. Utilizado para el desarrollo web y aplicaciones móviles híbridas con Onsen UI; implementa Javascript como lenguaje de programación, sin embargo, soporta Typescript.

Características:

- Curva de aprendizaje pequeña
- Virtual DOM
- Plantilla basada en HTML
- Reactividad
- Componentes reutilizables
- Enrutamiento
- Integraciones

Ventajas:

- Biblioteca súper pequeña. Apenas pesa 8KB después de Gzipping.
- Los desarrolladores pueden separar la plantilla del DOM virtual compilador e incluso el tiempo de ejecución.
- Supera los marcos voluminosos de Angular.
- Facilidad de comprensión y desarrollo.
- Más fácil de depurar, ahorrando mucho tiempo a los desarrolladores.
- Útil para crear aplicaciones completas o incluso reemplazar aplicaciones existentes parcialmente.
- La implementación de Virtual DOM lo convierte en una de las bibliotecas del lado de la cliente más rápida.

Desventajas:

- El desarrollo es hecho por una comunidad cerrada que es predominantemente chino provocando barreras idiomáticas.
- El desarrollador tiene que soportar la responsabilidad de un código más limpio y mantenible[9].

Spring

Es un framework modular de aplicación, open source y ligero, orientado a aplicaciones escritas en lenguaje de programación Java.

Características:

- Arquitectura divida en siete capas: Spring Core, Spring Context, Spring AOP (Aspect-oriented programming), Spring ORM, Spring DAO (Data Access Object), Spring Web, Spring Web MVC.[10]
- Posee documentación con todos los proyectos relacionados a su tecnología.
- Permite el desarrollo de aplicaciones flexibles, cohesivas y con bajo acoplamiento.
- Simplifica el desarrollo JEE (Java Enterprise Edition) por medio de la utilización de clases Java Simples para configurar los servicios.[11]

Laravel

Es un framework PHP orientado a crear código simple y elegante bajo el patrón de arquitectura MVC, actualmente cuenta con versiones LTS (Long Term Support); permite la creación de aplicaciones que puedan ser ejecutadas por consola y aplicaciones web.

Características:

- Curva de aprendizaje alta.
- Documentación completa, simple y gratuita.
- Gran comunidad de desarrolladores.
- Proyecto más popular en github desarrollado con PHP.
- Provee un conjunto de servicios y herramientas de infraestructura que facilitan su puesta en función en diferentes entornos, como Forge y Homestead.
- Brinda ORM, Eloquent, basado en el patrón active record.
- Utiliza un sistema de plantillas con un sistema de caché.
- Desarrollo de backend.[12]

Express.js

Tambien llamado Express, es un framework de aplicación web gratuito para Node.js lo que permite desarrollar aplicaciones web del lado del servidor con JavaScript y proporciona una estructura similar a MVC para las aplicaciones web.[13]

Características:

- Desarrollo de web REST de forma sencilla.[14]
- Flexible ya que posee numerosos módulos en npm con los que puede conectarse.
- Curva de aprendizaje baja
- Documentación abundante y gratuita.[15]

1.2.2.4. Aplicación web

Surgió durante la etapa de la web 1.0 durante la década de 1990 debido a las primeras conexiones de acceso conmutado y de las etiquetas multimedia del estándar HTML (Lenguaje de marcado de hipertexto o al inglés HyperText Markup Language) y la incorporación de pequeños programas desarrollados en java denominados applets, este tipo de aplicaciones cumple con la arquitectura constituida por máquinas conectadas a una red (Internet o Intranet) cuyo esquema se denomina cliente-servidor[16], razón por la cual se la puede definir como el software que reside en un ordenador conocido como servidor web. Dentro de las aplicaciones web se pueden encontrar diversos tipos, por ejemplo: gestores de correo, web mails, blogs y tiendas en línea; a su vez se pueden clasificar por el tipo de acceso, las cuales pueden ser: públicas y restringidas.[17]

1.2.2.5. Deporte

Es un hecho social e institucional proveniente del subconjunto del conjunto de prácticas motrices de carácter competitivo, teniendo en cuenta sus orígenes y que es una creación social, fechada y situada históricamente y dependiente del contexto en que se realiza también se lo define como el conjunto de medios que se utilizan para pasar tiempo agradable o a su vez como ejercicios físicos[18]; en base a la diversidad de manifestaciones se pueden distinguir dos tipos de deportes, tales como:

- Deporte de espectáculo, destinado a ser programado y explotado para su difusión y utilizado por las finanzas, industria y comercio.
- Deporte de ocio, considerado el primer deporte y ajeno a la comercialización, propaganda e instrumentalización por ser un deporte espontáneo.[19]

1.2.2.6. Selección deportiva

Es el proceso organizado cuyo objetivo es escoger por medio de métodos, técnicas y pruebas a los deportistas más capaces y dotados de aptitudes, basado en el control, análisis y valoración de una serie de factores que determinan el rendimiento, siendo estos las características o rasgos que pueden clasificarse en estables (no perfectibles) y lábiles (perfectibles).

Para la selección de talentos existen varios modelos de selección, por ejemplo: selección natural, selección técnica y selección científica. La selección natural se basa en una selección al azar debido a que el individuo es quien decide participar en el deporte ya sea por una tradición escolar o por deseo de los padres.

La selección técnica se centra en las cualidades técnicas del deportista debido a que es llevada a cabo por los técnicos deportivos quienes toman en cuenta la clase, condición física, grado de entrenamiento, forma y rendimiento óptimo durante la selección. La selección científica se basa en fundamentos científicos de las ciencias aplicadas como algunas características biológicas, psicológicas, antropométricas, físicas, entre otras.

Existen otros modelos como: el basado en la performance, en el cual la selección se da a través de los resultados en una única oportunidad, en la que llevan a cabo una serie de pruebas en las cuales se eliminan a ciertos participantes que no cumplen con lo establecido y seleccionan a los de mejor resultado, modelos procesuales que establecen estapas para la selección definitiva de los talentos deportivos y la detección por medio de filtros previos y seguimiento de la progresión uniendo los modelos procesuales y los basados en la performance.

Etapas en la selección de talentos

- Primera etapa: determinación de perfiles descriptivos del deporte, establecer criterios de detección y selección y preparación de instrumentos para recolección de información.
- Segunda etapa: localización de la información existente y realización de un filtro previo.
- Tercera etapa: aplicación del programa de selección, análisis de datos y toma de decisiones, exposición de resultados.
- Cuarta etapa: selección final.

1.2.2.7. Talento deportivo

Zatsiorski, considera el talento deportivo como "una combinación de capacidades motoras y psicológicas, así como de las aptitudes anatomofisiológicas que crean un en conjunto la posibilidad potencial para el logro de altos resultados deportivos en un deporte concreto". Por otra parte, Hahn lo define como "una aptitud acentuada en una dirección que supera la medida normal, que todavía no está desarrollada completamente".[20]

1.2.2.8. El fútbol como deporte

El fútbol es un deporte táctico y de conjunto en el que participan diferentes actores como: jugadores, director técnico y árbitro; cada equipo esta conformado de un total de once jugadores, cada uno con sus habilidades las cuales determinan su rendimiento y posición de juego; estas habilidades son: físicas, deportivas y mentales.

Un futbolista debe dominar su cuerpo de forma fisica y mental asi como tambien el balón y el juego; estos tres aspectos condicionan las posibilidades de éxito de un jugador.[21]

El dominio del cuerpo comprende los aspectos físicos y psicológicos de un jugador; las capacidades y habilidades físicas básicas con las que trabaja un futbolista son; fuerza, resistencia, velocidad, flexibilidad, coordinación, equilibrio y agilidad.

El dominio del balón engloba la técnica individual (conducciòn, regate y tiro) y colectiva (control de pases, remate, golpe de cabeza y despeje) del mismo.

El dominio del juego comprende los momentos como son ataque-defensa y sus transiciones.

Los aspectos psicológicos de un jugador comprenden los valores que este debe tener como son: competir, actitud, perseverancia, ambición, disciplina y respeto.[22]

1.3. Objetivos

1.3.1. Objetivo General

Desarrollar un sitio web para la gestión del reclutamiento y selección de futbolistas en la ciudad de Ambato.

1.3.2. Objetivos Específicos

- Analizar los requerimientos del sitio web para la selección y reclutamiento de futbolistas.
- Seleccionar un framework de desarrollo que se adapte a los requisitos del sistema.
- Implantar el sitio web para la gestión del reclutamiento y selección de futbolistas.

CAPÍTULO II

METODOLOGÍA

2.1. Materiales

Para el desarrollo del presente proyecto se utilizó un banco de preguntas y una ficha de observación.
Las técnicas de recolacción aplicadas fueron la entrevista y observación de campo para los cuales se diseño como instrumentos un banco de preguntas y una ficha de observación.
El banco de preguntas fue deseñado de la siguiente manera:

Universidad Técnica de Ambato

Facultad de Ingeniería en Sistemas, Electrónica e Industrial

Carrera de Ingeniería en Sistemas Computacionales e Informáticos

Entrevistador: Génesis Noemi Rubira Ramírez
Entrevistados:

Tabla 2.1: Entrevistados y cargo

Entrevistado	**Cargo**
	Vicepresidente del club deportivo Macará
	Coordinador general del club deportivo Técnico Universitario
	Coordinador general del club deportivo Mushuc Runa

Fuente: Elaborado por el investigador

Tabla 2.2: Cuadro de preguntas y respuestas

Preguntas	**Respuestas**
¿Antes de la pandemia, cuál era el proceso a seguir para seleccionar y reclutar nuevos futbolistas?	
¿Considera que la pandemia ha afectado a los clubes deportivos y como lo ha hecho?	
¿Durante la pandemia realizaron eventos para seleccionar nuevos futbolistas? Si, no y por qué	
Durante la pandemia, ¿Qué medidas preventivas se tomaron a la hora de reclutar nuevos talentos?	
¿Dichas medidas preventivas afectaron el proceso de selección y reclutamiento?	
¿Cómo afectaron las medidas preventivas al proceso de reclutamiento y selección?	
¿En comparación a procesos de reclutamiento antes de la pandemia podría decir que la cantidad de participantes durante la pandemia menoró o aumentó?	
¿En caso de tener acceso a una plataforma que permita reclutar jugadores con base a las habilidades exhibidas en un video, usted como dirigente la utilizaría?	

Fuente: Elaborado por el investigador

Universidad Técnica de Ambato

Facultad de Ingeniería en Sistemas, Electrónica e Industrial

Carrera de Ingeniería en Sistemas Computacionales e Informáticos

Autor: Génesis Noemi Rubira Ramírez

Ficha de observación

Objetivo: realizar visitas a los clubes deportivos para obtener una visión general del proceso que se lleva a cabo a la hora de seleccionar y reclutar futbolistas.

Escala de valoración

5= excelente, 4 = muy bueno, 3 = bueno, 2 = regular, 1 = insuficiente

Tabla 2.3: Ficha de observación

Descripción	5	4	3	2	1	Observación
Registro de datos de los postulantes						
Clasificación por sub y posición de juego de los postulantes						
Creación de equipos para los partidos de prueba						
Preselección de participantes						
Registro de observaciones de los jugadores						
Partido de prueba con los preseleccionados						
Selección final						
Registro de jugadores seleccionados						

Fuente: Elaborado por el investigador

2.2. Métodos

2.2.1. Modalidad de la Investigación

La modalidad de investigación es bibliográfica-documental.

Investigación bibliográfica

La presente investigación es bibliográfica porque utiliza fuentes como libros, documentos, artículos, revistas y tesis para la construcción del marco teórico.

Investigación documental

La presente investigación es documental porque se analiza el problema y los resultados de la investigación con el fin de proponer una solución a dicho problema.

2.2.2. Población y Muestra

La población para la presente investigación está conformada por 3 directivos de los principales clubes de fútbol de la ciudad de Ambato.

La presente investigación no requiere la obtención de una muestra debido a que la población con la que se va a trabajar es menor de 100 individuos.

2.2.3. Recolección de Información

Al aplicar la entrevista a los directivos de los principales equipos de fútbol de la ciudad de Ambato, se obtuvo los siguientes resultados.

Preguntas	**Respuestas**
¿Antes de la pandemia, cuál era el proceso a seguir para seleccionar y reclutar nuevos futbolistas?	Seleccionan a los jugadores de sus divisiones formativas tomando encuenta los logros, trayectoria, currículum del jugador e informe de los directores técnicos o por recomendación
¿Considera que la pandemia ha afectado a los clubes deportivos y como lo ha hecho?	Sí, las consecuencias fueron en parámetros considerables principalmente en el presupuesto del club
¿Durante la pandemia realizaron eventos para seleccionar nuevos futbolistas? Si, no y por qué	No porque durante la pandemia hubo muchas restricciones dentro del club como de los organismos de los torneos.
Durante la pandemia, ¿Qué medidas preventivas se tomaron a la hora de reclutar nuevos talentos?	No realizaron pruebas a ningún jugador.
¿Dichas medidas preventivas afectaron el proceso de selección y reclutamiento?	Sí
¿Cómo afectaron las medidas preventivas al proceso de reclutamiento y selección?	No se pudo reclutar nuevos jugadores.
¿En comparación a procesos de reclutamiento antes de la pandemia podría decir que la cantidad de participantes durante la pandemia menoró o aumentó?	Menoró
¿En caso de tener acceso a una plataforma que permita reclutar jugadores con base a las habilidades exhibidas en un video, usted como dirigente la utilizaría?	Sí

Tabla 2.4: Cuadro de entrevista a vicepresidente del club deportivo Macará

Fuente: Elaborado por el investigador

Preguntas	**Respuestas**
¿Antes de la pandemia, cuál era el proceso a seguir para seleccionar y reclutar nuevos futbolistas?	Hacen el llamado por redes sociales para partidos comprobatorios en lugares específicos.
¿Considera que la pandemia ha afectado a los clubes deportivos y como lo ha hecho?	Ha afectado al club y a los trabajadores en el ámbito económico ya que el presupuesto se redujo.
¿Durante la pandemia realizaron eventos para seleccionar nuevos futbolistas? Si, no y por qué	Durante la pandemia no se realizó ningún partido comprobatorio ya que estaba prohibido la aglomeración de personas.
Durante la pandemia, ¿Qué medidas preventivas se tomaron a la hora de reclutar nuevos talentos?	No se reclutó nuevos jugadores sin embargo las medidas en general que se tomaron fue la vacunación, y desinfección de cada miembro del club.
¿Dichas medidas preventivas afectaron el proceso de selección y reclutamiento?	Sí
¿Cómo afectaron las medidas preventivas al proceso de reclutamiento y selección?	No se pudo hacer ningún evento por lo cual no se pudo reclutar jugadores.
¿En comparación a procesos de reclutamiento antes de la pandemia podría decir que la cantidad de participantes durante la pandemia menoró o aumentó?	Menoró
¿En caso de tener acceso a una plataforma que permita reclutar jugadores con base a las habilidades exhibidas en un video, usted como dirigente la utilizaría?	Para seleccionar jugadores de equipos juveniles no sin embargo para el equipo de primera si lo utilizarián

Tabla 2.5: Cuadro de entrevista a coordinador general del club deportivo Técnico Universitario

Fuente: Elaborado por el investigador

Preguntas	Respuestas
¿Antes de la pandemia, cuál era el proceso a seguir para seleccionar y reclutar nuevos futbolistas?	Manejan algunos programas donde pueden ver los partidos en que los jugadores de su interés participan. No se pudo hacer ningún tipo de prueba hasta después de la pandemia
¿Considera que la pandemia ha afectado a los clubes deportivos y como lo ha hecho?	La pandemia afectó al club principalmente en el ámbito ecónomico ya que se maneja el presupuesto a base de taquillas y al no haber mucha asistencia de las personas a los escenarios deportivos los ingresos son menos.
¿Durante la pandemia realizaron eventos para seleccionar nuevos futbolistas? Si, no y por qué	No se pudo hacer ningún tipo de prueba hasta después de la pandemia
Durante la pandemia, ¿Qué medidas preventivas se tomaron a la hora de reclutar nuevos talentos?	No se realizó ninguna prueba pero después de la pandemia si y las medidas tomadas fueron: desinfección, mantener la distancia, uso de mascarillas y llevar la prueba pcr en negativo
¿Dichas medidas preventivas afectaron el proceso de selección y reclutamiento?	Sí
¿Cómo afectaron las medidas preventivas al proceso de reclutamiento y selección?	Afectaron en cuestión de tiempo y dinero
¿En comparación a procesos de reclutamiento antes de la pandemia podría decir que la cantidad de participantes durante la pandemia menoró o aumentó?	Disminuyó ya que habián personas con pocos recursos económicos y no podían realizarse las pruebas PCR que eran un requisito
¿En caso de tener acceso a una plataforma que permita reclutar jugadores con base a las habilidades exhibidas en un video, usted como dirigente la utilizaría?	Sí

Tabla 2.6: Cuadro de entrevista a coordinador general del club deportivo Mushuc Runa

Fuente: Elaborado por el investigador

En base a las entrevistas realizadas a los directivos de los principales clubes de fútbol de Ambato se pudo notar que a pesar de que cada club tiene un proceso

diferente para seleccionar y reclutar nuevos jugadores los tres se basan en evaluar su desempeño. También se pudo ver que la pandemia afectó de muchas formas a cada club principalmente en el ámbito económico y en la selección de nuevos talentos ya que no pudieron realizar ningún tipo de prueba durante este tiempo. Los tres directivos coincidieron en que si utilizarían un sistema que les permita visualizar las habilidades de un jugador por medio de videos para seleccionar y reclutar futbolistas.

De la observación de campo aplicada en el club deportivo Mushuc Runa se obtuvo como resultado la siguiente información:

Tabla 2.7: Ficha de observación

Descripción	**5**	**4**	**3**	**2**	**1**	**Observación**
Registro de datos de los postulantes				x		Registro de forma manual en fichas.
Clasificación por sub y posición de juego de los postulantes			x			Se agrupan a los jugadores por edades y posiciones de juego.
Creación de equipos para los partidos de prueba			x			De la clasificación anterior seleccionan 11 jugadores de las diferentes posiciones de juegos.
Preselección de participantes		x				Realizan un partido de 45 minutos con los jugadores antes seleccionados.
Registro de observaciones de los jugadores				x		El registro lo hacen de forma manual en la fichas de cada jugador durante el parido de preselección.
Partido de prueba con los preseleccionados		x				Realizan un partido de 45 minutos con los jugadores seleccionados del primer partido.
Selección final		x				Realizan un partido de 90 minutos con los jugadores seleccionados del segundo partido.
Registro de jugadores seleccionado				x		Guardan las fichas de los jugadores seleccionados en un archivero

Fuente: Elaborado por el investigador

Al realizar la visita al club deportivo Mushuc Runa se pudo observar que todos los procesos lo realizan de forma manual por medio de fichas y partidos de prueba, al igual que el archivo de la información de los jugadores seleccionados.

2.2.4. Procesamiento y Análisis de Datos

En base a la entrevista realizada a los principales clubes deportivos de la ciudad de Ambato y la observación de campo realizada al club deportivo Mushuc Runa se puede afirmar que:

- La cantidad de personas que llegan a probarse es muy grande y al llevar todos los procesos de forma manual el tiempo empleado en cada registro es extenso lo cual limita el tiempo de pruebas.
- Durante el proceso de selección se realizan tres partidos comprobatorios dos de 45 con el fin de reducir el número de participantes y uno de 90 minutos para realizar la selección final.
- Una vez que han sido seleccionados los jugadores sus fichas son almacenadas en archiveros y respaldados en un archivo de Excel.

CAPÍTULO III

RESULTADOS Y DISCUSIÓN

3.1. Análisis y discusión de resultados

3.1.1. Proceso de selección y reclutamiento de futbolistas

El proceso de selección y reclutamiento de futbolistas consta de tres etapas las cuales corresponden a 2 partidos comprobatorios de 45 minutos, en los que los directores técnicos observan el desempeño de cada participante en su posición de juego y van haciendo anotaciones en las fichas de cada jugador que tenga un buen rendimiento para seleccionarlo y que participe en el partido final de 90 minutos en el que realizan la selección final de los jugadores que van a formar parte de sus equipos para separar sus fichas y almacenarlas en archiveros.

3.1.2. Selección de la tecnología para el front-end

Para el front-end del presente proyecto se seleccionó un framework de desarrollo ya que facilita y agilita el proceso de desarrollo, para lo cual se realizó un cuadro comparativo entre tres frameworks; donde se analizan varios aspectos, como son: curva de aprendizaje, lenguaje de programación, patrón de diseño, utilización, lenguaje en el que está basado, representación en el navegador, arranque y enlace de datos. En el cuadro 3.1 se puede visualizar la información detallada de la comparación entre los distintos frameworks seleccionados.

Tabla 3.1: Comparación frameworks de desarrollo para frontend.

	Angular js	**React js**	**Vue js**
Curva de aprendizaje	Alta Documentación completa, simple y gratuita	Media Documentación detallada	Baja Documentación detallada
Basado en	JavaScript	JavaScript	JavaScript
Lenguaje de programación	TypeScript	JavaScript, soporta TypeScript	JavaScript, soporta TypeScript
Utilización	Desarrollo web y aplicaciones móviles híbridas	Desarrollo web y aplicaciones móviles nativas	Desarrollo web y aplicaciones móviles híbridas
Patrón de diseño	MVC poco definido	MVC	MVVM o Model-View-ViewModel
Representacion del navegador	Shadow Dom	Virtual Dom	Virtual Dom
Arranque	Angular CLI	CRA (Create React App)	Vue CLI
Enlace de datos	Bidireccional	Unidireccional	Bidireccional

Fuente: Elaborado por el investigador

En base al análisis del cuadro 3.1 se seleccionó como framework de desarrollo a Angular js, ya que a pesar de tener una curva de aprendizaje mas elevada que react y vue posee una documentación mucho mas detallada lo que que permite un mejor aprendizaje, también tiene una gran comunidad de desarrolladores lo que sirve para encontrar soluciones a cualquier problema que se va generando durante el desarrollo y a diferencia de los frameworks con los que se comparó que también poseen componentes pero al ser demasiados y es responsabilidad del desarrollador probar cada uno de ellos y ver cual se ajusta o funciona según sus necesidades, Angular posee componentes ya funcionales que facilitan el diseño y desarrollo del sitio web.

3.1.3. Selección de la tecnología para el back-end

Para el desarrollo del servidor se realizó un cuadro comparativo entre tres frameworks orientados al backend, donde se comparó la curva de aprendizaje, documentación, leguaje de programación, arquitectura y ciertas características.

Esta información se la puede observar de forma más detallada en el cuadro 3.3.

Tabla 3.3: Comparación frameworks de desarrollo para backend.

	Laravel	**Express.js**	**Spring**
Curva de aprendizaje	Alta Documentación completa, simple y gratuita	Baja Documentación abundante y gratuita	Alta Poca documentación
Lenguaje de programación	PHP	JavaScript	Java
Arquitectura	MVC	Compatible con MVC	MVC
Características	Código abierto y gratuito	Código abierto y gratuito Desarrollo de web REST de forma sencilla Numerosos módulos	Código abierto y gratuito Gestión de transacciones

Fuente: Elaborado por el investigador

Para el back-end del presente proyecto se seleccionó el framework Express.js ya que al analizar el cuadro 3.3 se pudo observar que a comparación de Laravel y Spring posee mayor documentación lo cual facilita su aprendizaje y al ser un marco de trabajo para Node.js es muchas más veloz, adaptable y cuenta con una gran cantidad de librerías o módulos lo que a su vez facilita el desarrollo de apis REST.

3.1.4. Selección de la metodología de desarrollo

Para la selección de la metodología de desarrollo se realizó un cuadro comparativo entre tres metodologías ágiles mas usadas para el desarrollo de software, donde se compararon sus características. En el cuadro 3.5 de puede visualizar la información detallada de la comparación de las metodologías.

Tabla 3.5: Comparación metodologías de desarrollo ágil.

	Extreme programming	**Scrum**	**Kanban**
Características	Desarrollo iterativo e incremental	Trabajo por iteraciones (sprints)	Flujo continuo
	Historias de usuario	Teblero Scrum	Tablero Kanban
	Equipos de 2 a 10 personas	Equipos de 6 a 15 personas	Equipos de todos los tamaños
	Las tareas terminadas pueden ser modificadas	Las tareas terminadas no se retocan	Puede haber cambios en cualquier momento
	Programación o creación del producto	Administración del proyecto	Gestión y visualización del flujo de trabajo

Fuente: Elaborado por el investigador

Para el desarrollo del presente proyecto se seleccionó la metodología de desarrollo ágil exteme programming (XP) ya que a diferencia de las metodologías con las que se comparó en el cuadro 3.3 XP se basa en la programación o creación del producto de software por lo que aplica principios de desarrollo de software y a su vez al ser iterativo e incremental es flexible y adaptable a los cambios durante todo el proceso de desarrollo del producto y esta orientado a proyectos pequeños en los cuales utiliza las historias de usuario para gestionar los requerimientos de los usuarios.

3.2. Desarrollo de la propuesta

3.2.1. Fase 1: Planificación

3.2.1.1. Levantamiento de información

Para el levantamiento de información se utilizaron dos métodos de recolección, los cuales son la entrevista, aplicada a los directivos de los tres principales clubes de fútbol de Ambato y la observación de campo dirigida.. Por medio de estos métodos se ha determinado las necesidades sobre la gestión del proceso de selección y reclutamiento de los futbolistas.

3.2.1.2. Descripción del cliente

Los principales equipos de fútbol actualmente seleccionan a los futbolistas por medio de juegos comprobatorios en los cuales evalúan su desempeño y observan su talento y habilidades; para registrar la información lo hacen de forma manual con formularios de inscripciones llenados por los postulantes. En base a esta información se determinó que procesos deben ser automatizados por el sitio web, dando como resultados los siguientes módulos:

- **Registro de usuarios**
- **Gestión de equipos**
- **Gestión de pagos**
- **Gestión de videos**
- **Mensajería**

3.2.1.3. Definición de roles

Tabla 3.7: Definición de roles

Rol	Responsable	Función
Usuario	Directivos o representantes de clubes de fútbol de Ambato y aspirantes a futbolistas	Comprobar y validar las funcionalidades de cada proceso a ser implementado.
Programador	Génesis Noemi Rubira Ramírez	Desarrollar el sitio web.
Tester	Génesis Noemi Rubira Ramírez	Pruebas de aceptación.

Fuente: Elaborado por el investigador

3.2.1.4. Historias de usuario

Para desarrollar el presente proyecto se establecieron las siguientes historias de usuarios.

Tabla 3.8: Historia de usuario: Modelado de la base de datos

Historia de usuario			
Número:	H01	**Usuario:**	Desarrollador
Nombre:	Modelado de la base de datos		
Prioridad:	Alta	**Riesgo:**	Alto
Puntos estimados:	4	**Iteración asignada:**	1
Programador responsable:	Génesis Rubira		
Descripción:	Modelar la base de datos conforme a los requisitos del sitio web y que permita gestionar la información del mismo.		
Observación:	ninguna		

Fuente: Elaborado por el investigador

Tabla 3.9: Historia de usuario: Autenticación de usuarios

Historia de usuario			
Número:	H02	**Usuario:**	Desarrollador
Nombre:	Autenticación de usuarios		
Prioridad:	Alta	**Riesgo:**	Alto
Puntos estimados:	18	**Iteración asignada:**	1
Programador responsable:	Génesis Rubira		
Descripción:	Diseño de la ventana de inicio de sesión, la cual debe contener un formulario con los campos de correo y clave ademas de un botón para entrar a la cuenta. Diseño de la ventana registro de usuarios, esta ventana se conforma por los campos de nombre, apellido, teléfono, dirección, fecha de nacimiento, género, email, clave y un botón para enviar la información del formulario.		
Observación:	ninguna		

Fuente: Elaborado por el investigador

Tabla 3.10: Historia de usuario: Módulo administrador

Historia de usuario			
Número:	H03	**Usuario:**	Desarrollador
Nombre:	Módulo administrador		
Prioridad:	Alta	**Riesgo:**	Alto
Puntos estimados:	8	**Iteración asignada:**	1
Programador responsable:	Génesis Rubira		
Descripción:	El módulo administrador consta de dos ventanas; registro de equipos con un formulario, el cual contiene los campos de nombre, dirección, teléfono y serie y un botón para guardar esta información y la ventana registro de representantes o directivos de equipos con un formulario donde se digita la información del usuario directivo (nombre, apellido, teléfono, dirección y equipo).		
Observación:	ninguna		

Fuente: Elaborado por el investigador

Tabla 3.11: Historia de usuario: Módulo usuario

Historia de usuario			
Número:	H04	**Usuario:**	Desarrollador
Nombre:	Módulo usuario		
Prioridad:	Alta	**Riesgo:**	Alto
Puntos estimados:	8	**Iteración asignada:**	2
Programador responsable:	Génesis Rubira		
Descripción:	En el módulo usuario se muestra información de los equipos agrupados por series y un formulario para subir videos en el cual se requiere del archivo del video, título, descripción, series, el o los equipos para que los visualicen y una ventana flotante con la solicitud de pago para subir videos; también se visualiza otra ventana con el listado de los mensajes recibidos para responder a ellos y otra ventana en la que se observa la información del usuario y se la edita.		
Observación:	ninguna		

Fuente: Elaborado por el investigador

Tabla 3.12: Historia de usuario: Servicios REST

Historia de usuario			
Número:	H05	**Usuario:**	Desarrollador
Nombre:	Servicios REST		
Prioridad:	Alta	**Riesgo:**	Alto
Puntos estimados:	19	**Iteración asignada:**	2
Programador responsable:	Génesis Rubira		
Descripción:	Desarrollar los métodos post, put y get y métodos adicionales del api para el manejo de datos del sistema. Consumo del api desde el front-end.		
Observación:	ninguna		

Fuente: Elaborado por el investigador

Tabla 3.13: Historia de usuario: Módulo representante de equipo

Historia de usuario			
Número:	H06	**Usuario:**	Desarrollador
Nombre:	Módulo representante de equipo		
Prioridad:	Alta	**Riesgo:**	Alto
Puntos estimados:	14	**Iteración asignada:**	2
Programador responsable:	Génesis Rubira		
Descripción:	Desarrollar el módulo representante de equipo el cual permite a los usuarios directivos suscribirse a un plan para ver y buscar videos, además de enviar y recibir mensajes y editar su información personal.		
Observación:	ninguna		

Fuente: Elaborado por el investigador

Tabla 3.14: Historia de usuario: Método de pago

Historia de usuario			
Número:	H07	**Usuario:**	Desarrollador
Nombre:	Método de pago		
Prioridad:	Alta	**Riesgo:**	Alto
Puntos estimados:	15	**Iteración asignada:**	3
Programador responsable:	Génesis Rubira		
Descripción:	Desarrollo de la ventana del método de pago con la descripción de los equipos seleccionados, el precio, total a pagar y botón de pago.		
Observación:	Los usuarios que suben y ven videos realizarán pagos con paypal para poder realizar dichas acciones.		

Fuente: Elaborado por el investigador

Tabla 3.15: Historia de usuario: Integración de nuevas ventanas

Historia de usuario			
Número:	H08	**Usuario:**	Desarrollador
Nombre:	Integración de nuevas ventanas		
Prioridad:	Alta	**Riesgo:**	Alto
Puntos estimados:	27	**Iteración asignada:**	3
Programador responsable:	Génesis Rubira		
Descripción:	Desarrollar e integrar ventanas extras o faltantes para el correcto funcionamiento del sistema.		
Observación:	Ninguna		

Fuente: Elaborado por el investigador

Tabla 3.16: Historia de usuario: Edición de módulos

Historia de usuario			
Número:	H09	**Usuario:**	Desarrollador
Nombre:	Edición de módulos		
Prioridad:	Alta	**Riesgo:**	Alto
Puntos estimados:	20	**Iteración asignada:**	3
Programador responsable:	Génesis Rubira		
Descripción:	Modificación e integración de ventanas dentro de cada módulo del sistema.		
Observación:	ninguna		

Fuente: Elaborado por el investigador

Tabla 3.17: Historia de usuario: Cuenta bloqueada y contenido no autorizado

Historia de usuario			
Número:	H10	**Usuario:**	Desarrollador
Nombre:	Cuenta bloqueada y contenido no autorizado.		
Prioridad:	Alta	**Riesgo:**	Alto
Puntos estimados:	29	**Iteración asignada:**	4
Programador responsable:	Génesis Rubira		
Descripción:	Realizar las ventanas de cuenta bloqueada y contenido no autorizado; la ventana de cuenta bloqueada contiene la el mensaje de cuenta bloqueada por inactividad y un botón para reactivar su cuenta mientras que la ventana de contenido no autorizado contiene un mensaje indicando que el contenido no es accesible.		
Observación:	ninguna		

Fuente: Elaborado por el investigador

Tabla 3.18: Historia de usuario: Tareas programadas

Historia de usuario			
Número:	H11	**Usuario:**	Desarrollador
Nombre:	Tareas programadas		
Prioridad:	Alta	**Riesgo:**	Alto
Puntos estimados:	8	**Iteración asignada:**	4
Programador responsable:	Génesis Rubira		
Descripción:	Realizar las tareas programadas para desactivar cuenta de usuario inactivo y desactivar el estado de suscripción del usuario.		
Observación:	ninguna		

Fuente: Elaborado por el investigador

Tabla 3.19: Historia de usuario: Términos y condiciones de uso

Historia de usuario			
Número:	H12	**Usuario:**	Desarrollador
Nombre:	Términos y condiciones de uso		
Prioridad:	Alta	**Riesgo:**	Alto
Puntos estimados:	9	**Iteración asignada:**	4
Programador responsable:	Génesis Rubira		
Descripción:	Realizar la ventana de términos de uso y describir de forma clara y entendible a los usuarios los términos y condiciones de uso del sitio web.		
Observación:	ninguna		

Fuente: Elaborado por el investigador

Tabla 3.20: Historia de usuario: Control de errores

Historia de usuario			
Número:	H13	**Usuario:**	Desarrollador
Nombre:	Control de errores		
Prioridad:	Alta	**Riesgo:**	Alto
Puntos estimados:	13	**Iteración asignada:**	5
Programador responsable:	Génesis Rubira		
Descripción:	Revisar e integrar controles de tipos de datos y demás controles necesarios en los formularios del sistema.		
Observación:	ninguna		

Fuente: Elaborado por el investigador

3.2.1.5. Tareas

Una vez establecidas las historias de usuarios y sus iteraciones, se definieron las tareas a realizarse para el desarrollo del producto.

Tabla 3.21: Tarea: Definición de entidades y relaciones

Tarea			
Número de tarea:	T01	**Código de historia:**	H01
Nombre de la tarea:	Definición de entidades y relaciones		
Tipo de tarea:	Diseño	**Puntos estimados:**	2
Fecha inicio:	07/04/2022	**Fecha fin:**	08/04/2022
Responsable:	Génesis Rubira		
Descripción:	Definir las tablas (entidades) con los datos correspondientes a cada una y sus relaciones correspondientes para el manejo de la información del sistemas.		

Fuente: Elaborado por el investigador

Tabla 3.22: Tarea: Creación de la base de datos

Tarea			
Número de tarea:	T02	**Código de historia:**	H01
Nombre de la tarea:	Creación de la base de datos		
Tipo de tarea:	Diseño	**Puntos estimados:**	2
Fecha inicio:	09/04/2022	**Fecha fin:**	10/04/2022
Responsable:	Génesis Rubira		
Descripción:	Crear la base de datos en el motos de base de datos con las entidades y relaciones antes definidas.		

Fuente: Elaborado por el investigador

Tabla 3.23: Tarea: Diseño de ventana 1. Inicio de sesión

Tarea			
Número de tarea:	T03	**Código de historia:**	H02
Nombre de la tarea:	Diseño de ventana 1. Inicio de sesión		
Tipo de tarea:	Diseño	**Puntos estimados:**	9
Fecha inicio:	11/04/2022	**Fecha fin:**	19/04/2022
Responsable:	Génesis Rubira		
Descripción:	Realizar el diseño de la interfaz para la ventana de inicio de sesión, la cual consta de un formulario en el cual el usuario debe digitar su correo y clave para acceder al sistema.		

Fuente: Elaborado por el investigador

Tabla 3.24: Tarea: Diseño de ventana 2. Registro de usuario

Tarea			
Número de tarea:	T04	**Código de historia:**	H02
Nombre de la tarea:	Diseño de ventana 2. Registro de usuario		
Tipo de tarea:	Diseño	**Puntos estimados:**	9
Fecha inicio:	20/04/2022	**Fecha fin:**	29/04/2022
Responsable:	Génesis Rubira		
Descripción:	Realizar el diseño de la interfaz para la ventana de registro de usuario, en el cual el usuario digite su información básica (nombre, apellido, teléfono, dirección, fecha de nacimiento, género, correo y clave) con la que se creará la cuenta.		

Fuente: Elaborado por el investigador

Tabla 3.25: Tarea: Diseño de ventana 3. Registro de equipos

Tarea			
Número de tarea:	T05	**Código de historia:**	H03
Nombre de la tarea:	Diseño de ventana 3. Registro de equipos		
Tipo de tarea:	Diseño	**Puntos estimados:**	4
Fecha inicio:	01/05/2022	**Fecha fin:**	04/05/2022
Responsable:	Génesis Rubira		
Descripción:	Realizar el diseño de la interfaz para la ventana de registro de equipos con los campos para el nombre del equipo, dirección, teléfono y serie de equipo (A o B).		

Fuente: Elaborado por el investigador

Tabla 3.26: Tarea: Diseño de ventana 4. Registro de representantes o directivo de los equipos

Tarea			
Número de tarea:	T06	**Código de historia:**	H03
Nombre de la tarea:	Diseño de ventana 4. Registro de representantes o directivo de los equipos		
Tipo de tarea:	Diseño	**Puntos estimados:**	4
Fecha inicio:	05/05/2022	**Fecha fin:**	08/05/2022
Responsable:	Génesis Rubira		
Descripción:	Realizar el diseño de la interfaz para la ventana de registro de representantes o directivos de los equipos con los campos para digitar el nombre, apellido, dirección, teléfono, email y clave del usuario directivo.		

Fuente: Elaborado por el investigador

Tabla 3.27: Tarea: Diseño de ventana flotante1. Perfil de usuario

Tarea			
Número de tarea:	T07	**Código de historia:**	H04
Nombre de la tarea:	Diseño de ventana flotante1. Perfil de usuario		
Tipo de tarea:	Diseño	**Puntos estimados:**	5
Fecha inicio:	09/05/2022	**Fecha fin:**	13/05/2022
Responsable:	Génesis Rubira		
Descripción:	Realizar el diseño de la interfaz para la ventana flotante para el perfil de usuario con la información general (nombre, apellido, dirección, teléfono, género, fecha de nacimiento y clave) que el usuario puede editar.		

Fuente: Elaborado por el investigador

Tabla 3.28: Tarea: Diseño de ventana 4. Inicio del módulo de usuario

Tarea			
Número de tarea:	T08	**Código de historia:**	H04
Nombre de la tarea:	Diseño de ventana 4. Inicio del módulo de usuario		
Tipo de tarea:	Diseño	**Puntos estimados:**	3
Fecha inicio:	13/05/2022	**Fecha fin:**	15/05/2022
Responsable:	Génesis Rubira		
Descripción:	Realizar el diseño de la interfaz para la ventana de inicio para el usuario, en la que se muestra la información de los equipos (nombre, dirección y precio) y el formulario para subir videos.		

Fuente: Elaborado por el investigador

Tabla 3.29: Tarea: Desarrollo del api

Tarea			
Número de tarea:	T09	**Código de historia:**	H05
Nombre de la tarea:	Desarrollo del api		
Tipo de tarea:	Desarrollo	**Puntos estimados:**	7
Fecha inicio:	16/05/2022	**Fecha fin:**	22/05/2022
Responsable:	Génesis Rubira		
Descripción:	Se desarrollarán los distintos métodos que brindarán los servicios para el manejo de la información del sistema, tales como: login, registro, editar usuario, registrar equipos, cargar serie de equipos y cargar equipos.		

Fuente: Elaborado por el investigador

Tabla 3.30: Tarea: Consumo del api

Tarea			
Número de tarea:	T10	**Código de historia:**	H05
Nombre de la tarea:	Consumo del api		
Tipo de tarea:	Desarrollo	**Puntos estimados:**	10
Fecha inicio:	23/05/2022	**Fecha fin:**	03/06/2022
Responsable:	Génesis Rubira		
Descripción:	Se conectarán los distintos métodos del api (login, registro, editar usuario, registrar equipo, cargar series y cargar equipos) dentro de cada ventana diseñada en Angular js.		

Fuente: Elaborado por el investigador

Tabla 3.31: Tarea: Captura de miniatura de video

Tarea			
Número de tarea:	T11	**Código de historia:**	H05
Nombre de la tarea:	Captura de miniatura de video		
Tipo de tarea:	Desarrollo	**Puntos estimados:**	2
Fecha inicio:	04/06/2022	**Fecha fin:**	05/06/2022
Responsable:	Génesis Rubira		
Descripción:	Desarrollar el método para capturar la miniatura de cada video subido por parte del usuario.		

Fuente: Elaborado por el investigador

Tabla 3.32: Tarea: Diseño de ventana 5. Listado de videos

Tarea			
Número de tarea:	T12	**Código de historia:**	H06
Nombre de la tarea:	Diseño de ventana 5. Listado de videos		
Tipo de tarea:	Diseño	**Puntos estimados:**	5
Fecha inicio:	06/06/2022	**Fecha fin:**	10/06/2022
Responsable:	Génesis Rubira		
Descripción:	Diseñar la interfaz en la que se listan los videos del directivo o representante del equipo, donde se visualice un buscador, una lista de filtros y la miniatura, título, descripción, nombre del usuario y fecha en que los videos fueron subidos.		

Fuente: Elaborado por el investigador

Tabla 3.33: Tarea: Consumo de método del api para listar videos.

Tarea			
Número de tarea:	T13	**Código de historia:**	H06
Nombre de la tarea:	Consumo de método del api para listar videos.		
Tipo de tarea:	Desarrollo	**Puntos estimados:**	2
Fecha inicio:	11/06/2022	**Fecha fin:**	12/06/2022
Responsable:	Génesis Rubira		
Descripción:	Consumir el método para obtener la información de los videos subidos de los usuarios y listarla en la pantalla de listado de videos.		

Fuente: Elaborado por el investigador

Tabla 3.34: Tarea: Diseño de ventana 6. Video

Tarea			
Número de tarea:	T14	**Código de historia:**	H06
Nombre de la tarea:	Tarea: Diseño de ventana 6. Video		
Tipo de tarea:	Diseño	**Puntos estimados:**	7
Fecha inicio:	13/06/2022	**Fecha fin:**	19/06/2022
Responsable:	Génesis Rubira		
Descripción:	Diseñar la interfaz para reproducir el video y visualizar la información del usuario propietario del video con un botón para contactarse con el usuario por medio de un mensaje.		

Fuente: Elaborado por el investigador

Tabla 3.35: Tarea: Configuración de cuenta developer de PayPal

Tarea			
Número de tarea:	T15	**Código de historia:**	H07
Nombre de la tarea:	Configuración de cuenta developer de PayPal		
Tipo de tarea:	Configuración	**Puntos estimados:**	5
Fecha inicio:	20/06/2022	**Fecha fin:**	24/06/2022
Responsable:	Génesis Rubira		
Descripción:	Crear y configurar una cuenta de tipo developer en PayPal para recibir pagos.		

Fuente: Elaborado por el investigador

Tabla 3.36: Tarea: Integración de PayPal en Angular

Tarea			
Número de tarea:	T16	**Código de historia:**	H07
Nombre de la tarea:	Integración de PayPal en Angular		
Tipo de tarea:	Desarrollo	**Puntos estimados:**	4
Fecha inicio:	25/06/2022	**Fecha fin:**	28/06/2022
Responsable:	Génesis Rubira		
Descripción:	Integrar el script de PayPal en el archivo index.html, configurar el id del cliente del script para realizar pagos.		

Fuente: Elaborado por el investigador

Tabla 3.37: Tarea: Diseño ventana 7 y 8. Forma de pago

Tarea			
Número de tarea:	T17	**Código de historia:**	H07
Nombre de la tarea:	Diseño ventana 7 y 8. Forma de pago		
Tipo de tarea:	Diseño	**Puntos estimados:**	6
Fecha inicio:	29/06/2022	**Fecha fin:**	05/07/2022
Responsable:	Génesis Rubira		
Descripción:	Diseñar las interfaces para la ventana de forma de pago; en la ventana 7 se visualiza la información de cada suscripción con un botón para realizar el pago y suscribirse y la ventana 8 corresponde al resumen de lo que el usuario debe cancelar con la descripción, precio, total y el botón de pago de PayPal.		

Fuente: Elaborado por el investigador

Tabla 3.38: Tarea: Diseño ventana 9. Habilidades

Tarea			
Número de tarea:	T18	**Código de historia:**	H08
Nombre de la tarea:	Diseño ventana 9. Habilidades		
Tipo de tarea:	Diseño	**Puntos estimados:**	6
Fecha inicio:	06/07/2022	**Fecha fin:**	12/07/2022
Responsable:	Génesis Rubira		
Descripción:	Diseñar ventana de habilidades donde se listen checkboxs con todas las habilidades que un usuario puede tener y un botón para guardarlas.		

Fuente: Elaborado por el investigador

Tabla 3.39: Tarea: Componente de mensaje

Tarea			
Número de tarea:	T19	**Código de historia:**	H08
Nombre de la tarea:	Componente de mensaje.		
Tipo de tarea:	Desarrollo	**Puntos estimados:**	8
Fecha inicio:	13/07/2022	**Fecha fin:**	22/07/2022
Responsable:	Génesis Rubira		
Descripción:	Diseñar la interfaz del componente para el envío de mensajes donde se visualice el campo para escribir el mensaje, el botón para el envío, el nombre del usuario receptor y un botón para cerrar el componente y consumir el servicio para realizar el envío.		

Fuente: Elaborado por el investigador

Tabla 3.40: Tarea: Diseño ventana 10. Listado de mensajes

Tarea			
Número de tarea:	T20	**Código de historia:**	H08
Nombre de la tarea:	Diseño ventana 10. Listado de mensajes		
Tipo de tarea:	Diseño	**Puntos estimados:**	4
Fecha inicio:	23/07/2022	**Fecha fin:**	28/07/2022
Responsable:	Génesis Rubira		
Descripción:	Diseñar la ventana de listado de mensajes, en la que en la parte izquierda se listan los nombres de los usuarios y el último mensaje enviado de cada uno y en la derecha todos los mensajes enviados y recibidos y un campo para redactar un nuevo mensaje y enviarlo.		

Fuente: Elaborado por el investigador

Tabla 3.41: Tarea: Integrar servicios en la ventana 9. Habilidades

Tarea			
Número de tarea:	T21	**Código de historia:**	H08
Nombre de la tarea:	Integrar servicios en la ventana 9. Habilidades		
Tipo de tarea:	Desarrollo	**Puntos estimados:**	6
Fecha inicio:	29/07/2022	**Fecha fin:**	05/08/2022
Responsable:	Génesis Rubira		
Descripción:	Consumir el método del api para obtener todos las habilidades y listarlas en la interfaz de Habilidades.		

Fuente: Elaborado por el investigador

Tabla 3.42: Tarea: Integrar servicios en la ventana 10. Listado de mensajes

Tarea			
Número de tarea:	T22	**Código de historia:**	H08
Nombre de la tarea:	Integrar servicios en la ventana 10. Listado de mensajes		
Tipo de tarea:	Desarrollo	**Puntos estimados:**	3
Fecha inicio:	05/08/2022	**Fecha fin:**	07/08/2022
Responsable:	Génesis Rubira		
Descripción:	Consumir los métodos de la api para obtener todos los chat-rooms, mensajes y enviar mensajes y listar los chat-rooms den la interfaz de listado de mensajes.		

Fuente: Elaborado por el investigador

Tabla 3.43: Tarea: Edición módulo usuario

Tarea			
Número de tarea:	T22	**Código de historia:**	H09
Nombre de la tarea:	Edición módulo usuario		
Tipo de tarea:	Desarrollo	**Puntos estimados:**	4
Fecha inicio:	08/08/2022	**Fecha fin:**	11/08/2022
Responsable:	Génesis Rubira		
Descripción:	Añadir campos para seleccionar la descripción y digitar el título del video ha subir en el formulario del módulo usuario.		

Fuente: Elaborado por el investigador

Tabla 3.44: Tarea: Edición ventana 5. Listado videos

Tarea			
Número de tarea:	T22	**Código de historia:**	H09
Nombre de la tarea:	Edición ventana 5. Listado videos		
Tipo de tarea:	Desarrollo	**Puntos estimados:**	3
Fecha inicio:	12/08/2022	**Fecha fin:**	14/08/2022
Responsable:	Génesis Rubira		
Descripción:	Añadir un buscador y filtros según las habilidades para el listado de videos.		

Fuente: Elaborado por el investigador

Tabla 3.45: Tarea: Ventana 11. Mis videos

Tarea			
Número de tarea:	T23	**Código de historia:**	H09
Nombre de la tarea:	Ventana 11. Mis videos		
Tipo de tarea:	Desarrollo	**Puntos estimados:**	8
Fecha inicio:	15/08/2022	**Fecha fin:**	24/08/2022
Responsable:	Génesis Rubira		
Descripción:	Desarrollar una nueva ventana en el módulo de usuario donde se pueda visualizar las miniaturas de los videos con sus descripción y fecha de subida y un buscador sus videos subidos.		

Fuente: Elaborado por el investigador

Tabla 3.46: Tarea: Carga de habilidades para la descripción del video

Tarea			
Número de tarea:	T23	**Código de historia:**	H09
Nombre de la tarea:	Carga de habilidades para la descripción de video		
Tipo de tarea:	Desarrollo	**Puntos estimados:**	5
Fecha inicio:	31/08/2022	**Fecha fin:**	06/09/2022
Responsable:	Génesis Rubira		
Descripción:	Consumir el método del api para obtener las habilidades del usuario y asignarla en el select para la descripción del video.		

Fuente: Elaborado por el investigador

Tabla 3.47: Tarea: Ventana 12. Cuenta bloqueada

Tarea			
Número de tarea:	T24	**Código de historia:**	H10
Nombre de la tarea:	Ventana 12. Cuenta bloqueada		
Tipo de tarea:	Desarrollo	**Puntos estimados:**	12
Fecha inicio:	07/09/2022	**Fecha fin:**	25/10/2022
Responsable:	Génesis Rubira		
Descripción:	Diseñar la ventana de cuenta bloqueada, con el mensaje de cuenta bloqueada por inactividad y un botón para activar la cuenta.		

Fuente: Elaborado por el investigador

Tabla 3.48: Tarea: Activación de cuenta

Tarea			
Número de tarea:	T25	**Código de historia:**	H10
Nombre de la tarea:	Activación de cuenta		
Tipo de tarea:	Desarrollo	**Puntos estimados:**	11
Fecha inicio:	30/09/2022	**Fecha fin:**	15/10/2022
Responsable:	Génesis Rubira		
Descripción:	Integrar una ventana modal para activar la cuenta por medio de un pago con PayPal donde indique el valor a pagar y el botón de pago de PayPal.		

Fuente: Elaborado por el investigador

Tabla 3.49: Tarea: Ventana 13. Contenido no autorizado

Tarea			
Número de tarea:	T26	**Código de historia:**	H10
Nombre de la tarea:	Ventana 13. Contenido no autorizado		
Tipo de tarea:	Desarrollo	**Puntos estimados:**	6
Fecha inicio:	15/10/2022	**Fecha fin:**	20/10/2022
Responsable:	Génesis Rubira		
Descripción:	Diseñar e integrará la ventana de contenido no autorizado con el mensaje de contenido no autorizado y un botón para redirigir al usuario a la página de inicio de sesión .		

Fuente: Elaborado por el investigador

Tabla 3.50: Tarea: Realizar tareas programadas

Tarea			
Número de tarea:	T27	**Código de historia:**	H11
Nombre de la tarea:	Realizar tareas programadas		
Tipo de tarea:	Desarrollo	**Puntos estimados:**	8
Fecha inicio:	21/10/2022	**Fecha fin:**	30/10/2022
Responsable:	Génesis Rubira		
Descripción:	Realizar los métodos en la api para la actualización del estado de la suscripción del usuario directivo y el estado del usuario por inactividad y configurar tareas programadas para ejecutarse diariamente a las 12 am y el 28 de cada mes a las 12 am, respectivamente.		

Fuente: Elaborado por el investigador

Tabla 3.51: Tarea: Ventana modal de términos y condiciones de uso

Tarea			
Número de tarea:	T28	**Código de historia:**	H12
Nombre de la tarea:	Ventana modal de términos y condiciones de uso		
Tipo de tarea:	Desarrollo	**Puntos estimados:**	9
Fecha inicio:	31/10/2022	**Fecha fin:**	10/11/2022
Responsable:	Génesis Rubira		
Descripción:	Desarrollar la ventana modal de términos y condiciones de uso e integrarla en el menú y ventana de registro de usuario.		

Fuente: Elaborado por el investigador

Tabla 3.52: Tarea: Revisión y control de errores

Tarea			
Número de tarea:	T29	**Código de historia:**	H13
Nombre de la tarea:	Revisión y control de errores		
Tipo de tarea:	Desarrollo	**Puntos estimados:**	13
Fecha inicio:	11/11/2022	**Fecha fin:**	29/11/2022
Responsable:	Génesis Rubira		
Descripción:	Revisar y controlar la duración del video subido, campos vacíos y tipos de datos en todos los formularios del sistema.		

Fuente: Elaborado por el investigador

3.2.1.6. Valoración de historias de usuario

Las historias de usuarios detallaron los requerimientos iniciales y adicionales del sistema, cada una conformada por valoraciones definidas lo que permitió definir que se trabajará un total de 4 horas diarias.

3.2.1.7. Estimación de historias de usuario

Se realizó la estimación del esfuerzo ha realizarse en las historias de usuario, que se representarán con un total de seis iteraciones según la funcionalidad, obteniendo de esta manera una aproximación de la duración del proyecto.

Tabla 3.53: Estimación de historias de usuario

Iteración	Número	Historia de usuario	Tiempo estimado	
			Días	Horas
1	H01	Modelado de la base de datos	4	16
1	H02	Autenticación de usuarios	18	72
1	H03	Módulo administrador	8	32
2	H04	Módulo usuario	8	32
2	H05	Servicios REST	19	76
2	H06	Módulo representante de equipo	14	56
3	H07	Método de pago	15	60
3	H08	Integración de nuevas ventanas	27	108
3	H09	Edición de módulos	20	80
4	H10	Cuenta bloqueada y contenido no autorizado	29	116
4	H11	Tareas programadas	8	32
4	H12	Términos y condiciones de uso	9	36
5	H13	Control de errores	13	52
Tiempo total de estimación (días/horas)			192	768

Fuente: Elaborado por el investigador

Iteración 1

Tabla 3.54: Resumen iteración 1

Iteración	Número	Historia de usuario	Tiempo estimado	
			Días	Horas
1	H01	Modelado de la base de datos	4	16
1	H02	Autenticación de usuarios	18	72
1	H03	Módulo administrador	8	32
Total			30	120

Fuente: Elaborado por el investigador

Iteración 2

Tabla 3.55: Resumen iteración 2

Iteración	Número	Historia de usuario	Tiempo estimado	
			Días	Horas
2	H04	Módulo usuario	8	32
2	H05	Servicios REST	19	76
2	H06	Módulo representante de equipo	14	56
Total			41	164

Fuente: Elaborado por el investigador

Iteración 3

Tabla 3.56: Resumen iteración 3

Iteración	Número	Historia de usuario	Tiempo estimado	
			Días	Horas
3	H07	Método de pago	15	60
3	H08	Integración de nuevas ventanas	27	108
3	H09	Edición de módulos	20	80
Total			62	248

Fuente: Elaborado por el investigador

Iteración 4

Tabla 3.57: Resumen iteración 4

Iteración	Número	Historia de usuario	Tiempo estimado	
			Días	Horas
4	H10	Cuenta bloqueada y contenido no autorizado	29	116
4	H11	Tareas programadas	8	32
4	H12	Términos y condiciones de uso	9	36
Total			46	184

Fuente: Elaborado por el investigador

Iteración 5

Tabla 3.58: Resumen iteración 5

Iteración	Número	Historia de usuario	Tiempo estimado	
			Días	Horas
5	H13	Control de errores	13	52
Total			13	52

Fuente: Elaborado por el investigador

3.2.1.8. Plan de entrega

Tabla 3.59: Plan de entrega

N°	Historia de usuario	Tiempo estimado		Iteración asignada					Entrega asignada				
		Días	Horas	1	2	3	4	5	1	2	3	4	5
001	Modelado de la base de datos	4	16	X					X				
002	Autenticación de usuarios	18	72	X					X				
003	Módulo administrador	8	32	X					X				
004	Módulo usuario	8	32		X					X			
005	Servicios REST	19	76		X					X			
006	Módulo representante de equipo	14	56		X					X			
007	Método de pago	15	60			X					X		
008	Integración de nuevas ventanas	27	108			X					X		
009	Edición de módulos	20	80			X					X		
010	Cuenta bloqueada y contenido no autorizado	20	116				X					X	
011	Tareas programadas	8	32				X					X	
012	Términos y condiciones de uso	9	36				X					X	
013	Control de errores	13	52					X					X

Fuente: Elaborado por el investigador

3.2.2. Fase 2: Diseño

3.2.2.1. Diseño de la base de datos

Como gestor de base de datos se utilizó phpMyAdmin en el cual se definieron las entidades con las que el sistema trabaja y son:
roles: esta tabla contiene la información de los distintos roles del sistema tales como: administrador, usuario y representante de equipo y se relaciona con la tabla usuarios.
usuarios: contiene información de todos los usuarios .
series: esta tabla tiene la información sobre las series para clasificar los equipos y se relaciona con la tabla equipos.
equipos: en esta tabla se guarda la información de los equipos.
suscripciones: esta tabla guarda información de todos los planes de suscripción que el sistema maneja.
habilidades: en esta tabla se guarda la información sobre las habilidades básicas de un fulbolista y se relaciona con la tabla usuario_habilidads
videos: en esta tabla se guarda la información de los videos subidos por los usuarios y el o los equipos para el que fue subido.
mensajes: esta tabla tiene la información del mensaje que los usuarios envían y el chat_room al que pertenecen.
chat_rooms: esta tabla contiene información sobre emisor y receptor de mensajes y está relacionada a la tabla mensajes.
equipo_videos: esta tabla surge de la relación de muchos a muchos entre las tablas equipos y videos.
usuario_habilidads: esta tabla surge de la relación de las tablas usuarios y habilidades, en esta tabla se guardan las habilidades que el usuario haya seleccionado.
usuario_suscripcions: esta tabla surge de la relación de muchos a muchos entre las tablas usuarios y suscripciones y se guarda la información del usuario y la suscripción seleccionada con su fecha de inicio y fin.

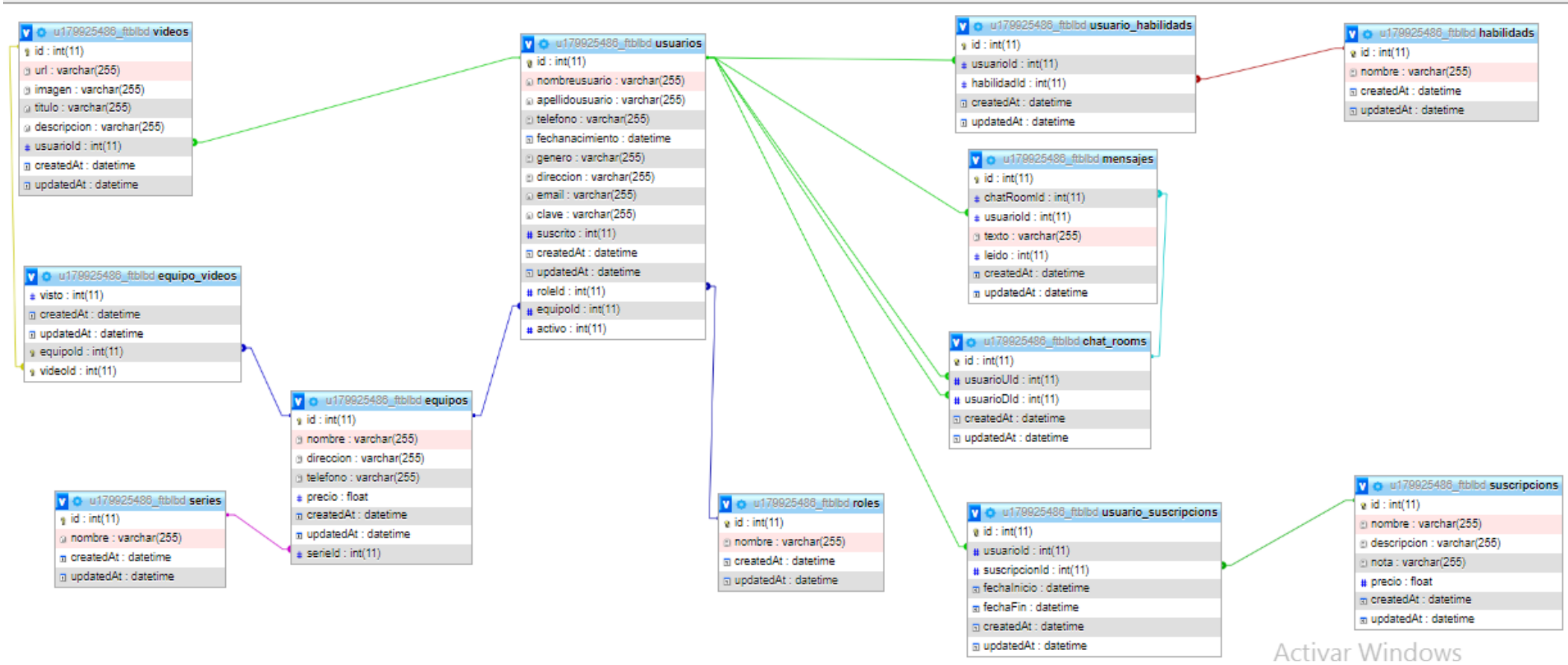


Figura 3.1: Modelado de la base de datos
Fuente: Elaborado por el investigador

3.2.2.2. Tarjetas CRC

Tabla 3.60: Tarjeta CRC - Usuario

Clase: Usuario	
Responsabilidades	**Colaboradores**
Registrar y editar usuario.	
Asignar rol de usuario.	
Asignar habilidades.	
Subir videos	
Realizar pagos	
Responder mensajes	Representante de equipo

Fuente: Elaborado por el investigador

Tabla 3.61: Tarjeta CRC - Usuario

Clase: Administrador	
Responsabilidades	**Colaboradores**
Registrar equipos.	
Registrar representantes de equipos.	
Asignar equipo a representante.	Representante de equipo

Fuente: Elaborado por el investigador

Tabla 3.62: Tarjeta CRC - Representante de equipo

Clase: Representante de equipo	
Responsabilidades	**Colaboradores**
Editar usuario.	
Asignar rol de usuario.	
Realizar pagos.	Productos.
Ver videos	
Enviar y recibir mensajes	Usuario

Fuente: Elaborado por el investigador

3.2.2.3. Diseño del sitio web

En la pantalla de registro de usuario el postulante a futbolista registra sus datos generales como son: nombre, apellido, teléfono, dirección, género, fecha de nacimiento, correo y clave para poder tener acceso al sistema.

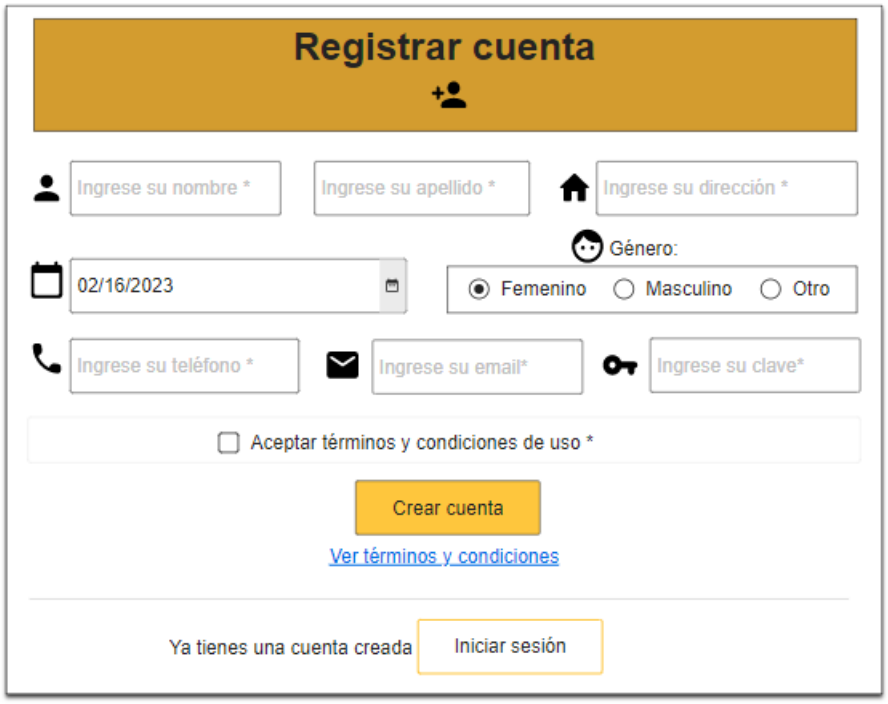


Figura 3.2: Interfaz de registro de usuarios
Fuente: Elaborado por el investigador

La pantalla de inicio de sesión tiene los campos de correo electrónico y clave para validar el acceso al sistema.

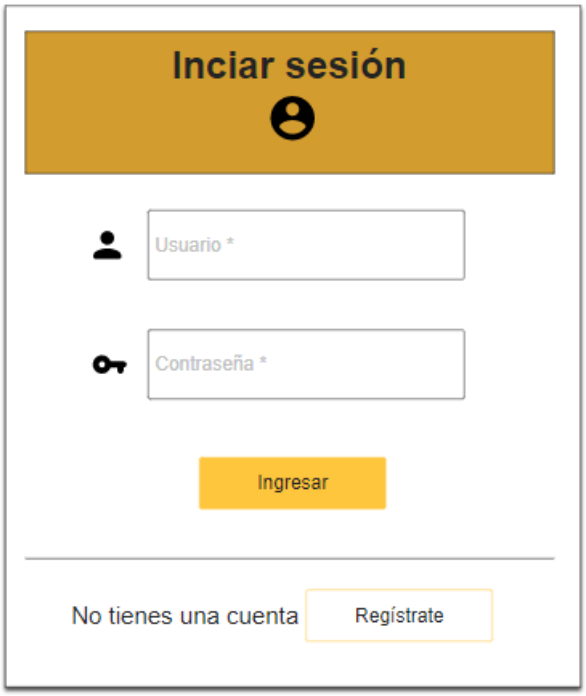


Figura 3.3: Interfaz inicio de sesión
Fuente: Elaborado por el investigador

El módulo de administrador esta conformado por dos pantallas o ventanas; registro de equipos y registro de usuario representante de equipo.
En la pantalla de registro de equipos se visualizan los campos de nombre, dirección, teléfono y un select de series para la creación de un nuevo equipo.

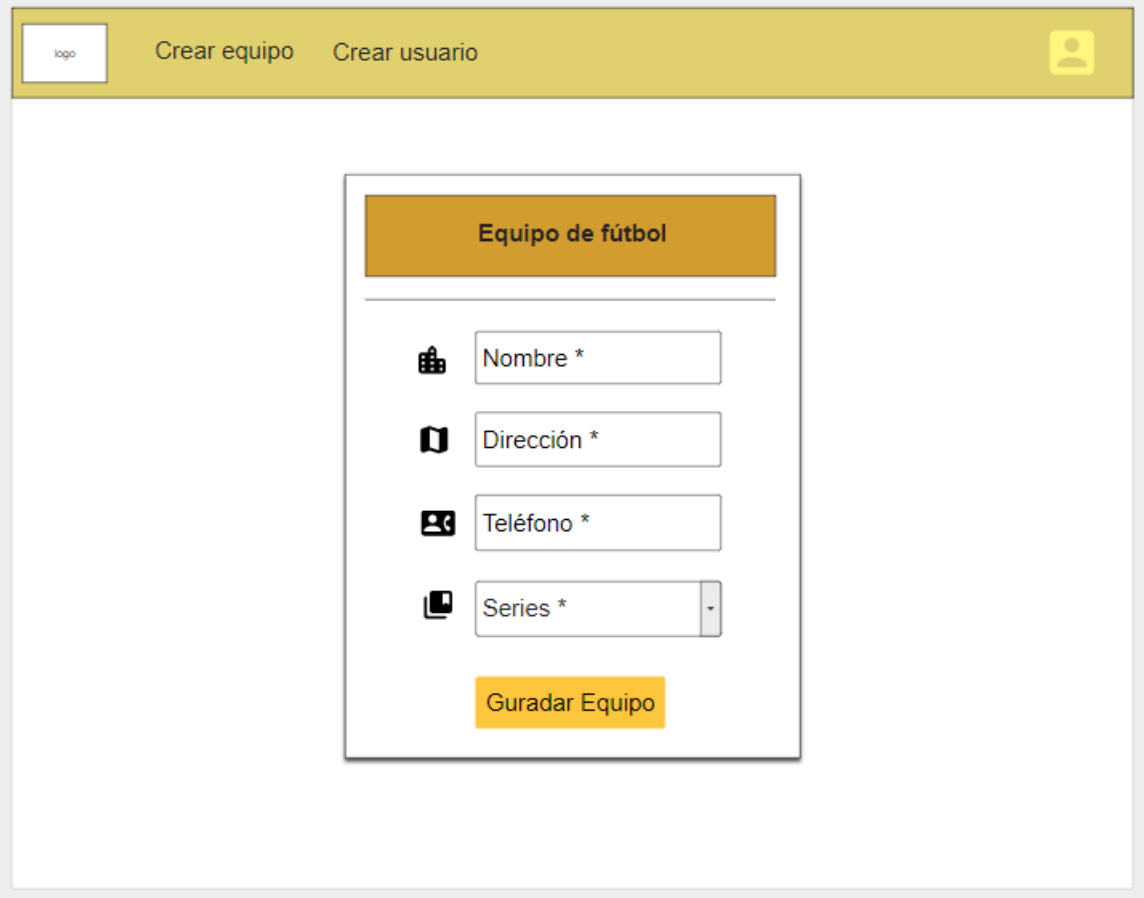


Figura 3.4: Interfaz registro de equipo
Fuente: Elaborado por el investigador

La pantalla de registro de usuario representante de equipo cuenta con los campos de nombre, apellido, dirección, teléfono, correo, clave y un select de los equipos existentes dentro del sistema para la creación de un nuevo usuario directivo.

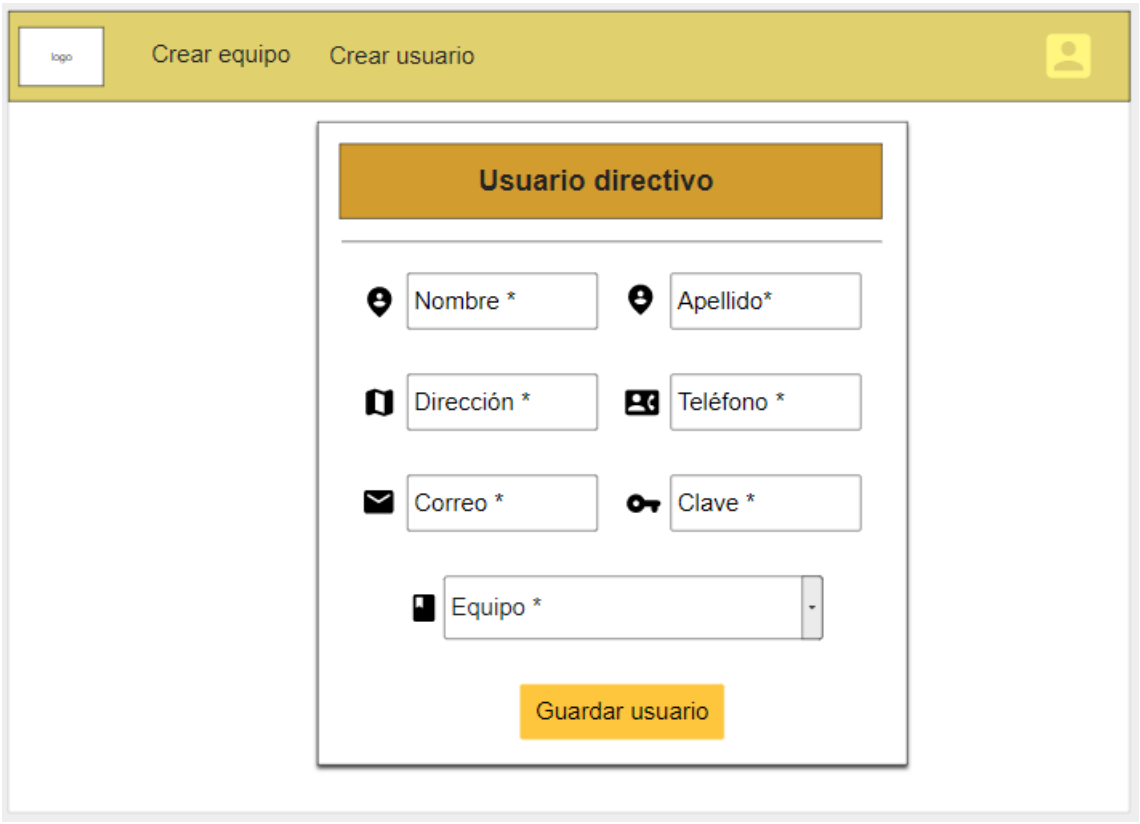


Figura 3.5: Interfaz registro de representante de equipo o directivo
Fuente: Elaborado por el investigador

El módulo de usuario cuenta con dos ventanas, la pantalla de inicio y la de mis videos.

La pantalla de inicio contiene la información de los equipos existentes agrupados por series y el formulario para subir el video.

Figura 3.6: Interfaz usuario
Fuente: Elaborado por el investigador

En la pantalla de mis videos se listan todos los videos que el usuario ha subido con nombre, título y fecha.

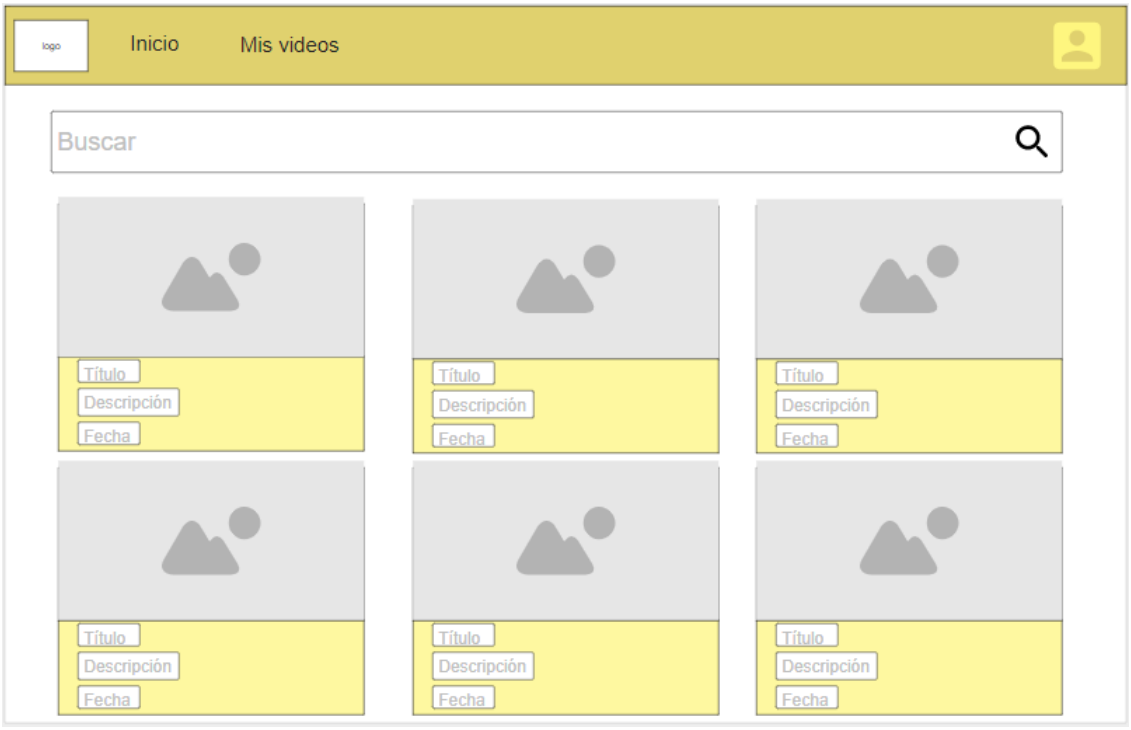


Figura 3.7: Interfaz mis videos
Fuente: Elaborado por el investigador

En la pantalla del representante de equipo se listan los videos que hayan sido subidos para su equipo.

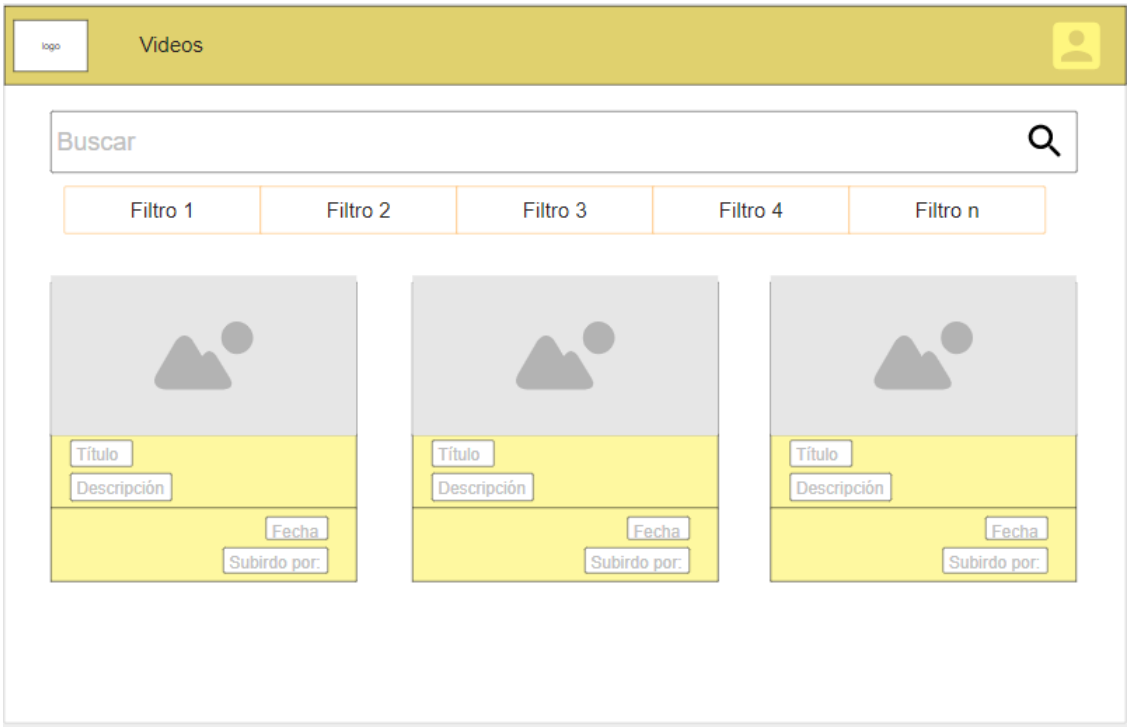


Figura 3.8: Interfaz listado videos
Fuente: Elaborado por el investigador

En la pantalla de listado de mensajes se observan todos los mensajes que el usuario ha recibido y la fecha en la que este fue enviado.

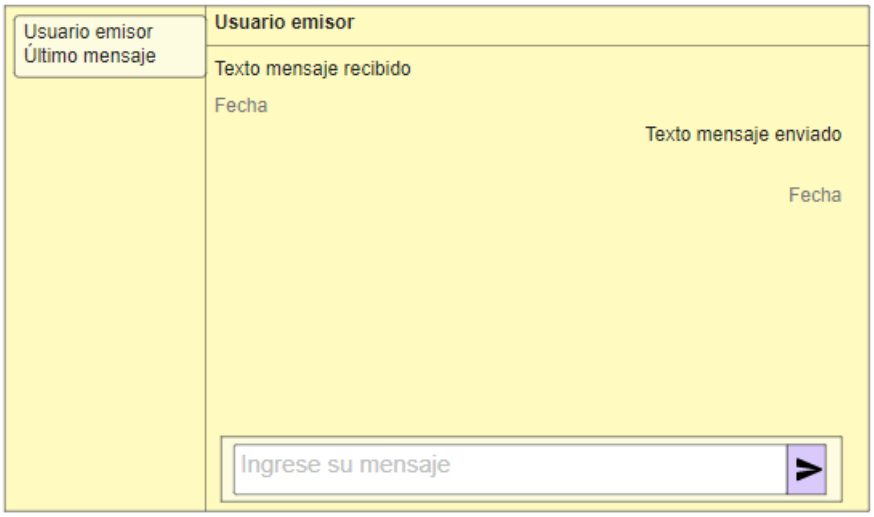


Figura 3.9: Interfaz listado de mensajes
Fuente: Elaborado por el investigador

La pantalla de perfil de usuario cuenta con los campos con los datos generales del usuario logueado.

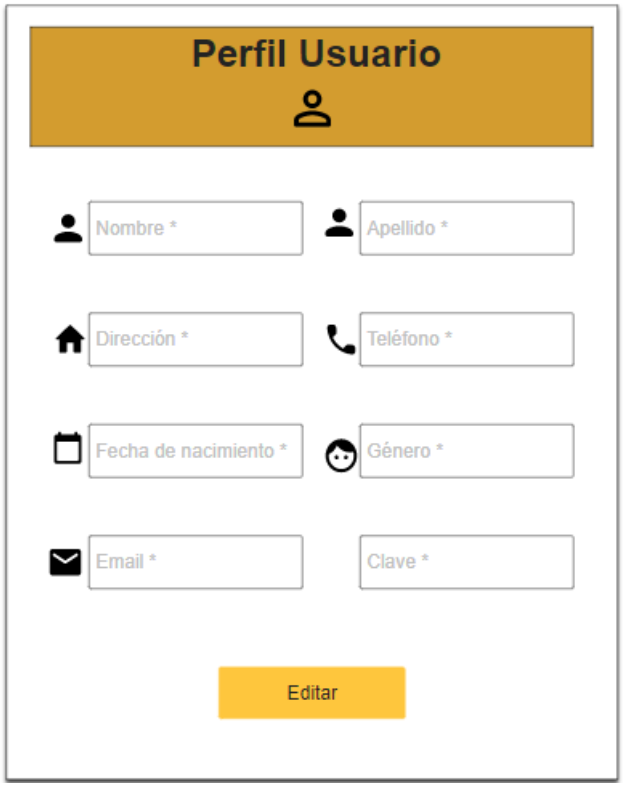


Figura 3.10: Interfaz mi perfil
Fuente: Elaborado por el investigador

La pantalla de contenido no autorizado muestra un corto mensaje indicando que debe iniciar sesión para interactuar con cada pantalla del sistema según el rol que posee.

Figura 3.11: Interfaz de contenido no autorizado
Fuente: Elaborado por el investigador

La pantalla de cuenta bloqueada aparece un pequeño mensaje de cuenta bloqueada y un botón para activar la cuenta.

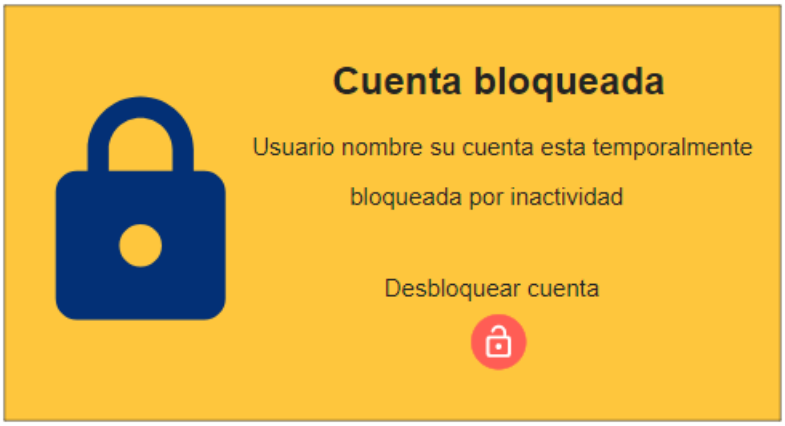


Figura 3.12: Interfaz de cuenta bloqueada
Fuente: Elaborado por el investigador

En la ventana modal de pago se muestra una tabla con el resumen de lo que el usuario va a pagar con su costo, total y el botón de paypal para realizar el pago.

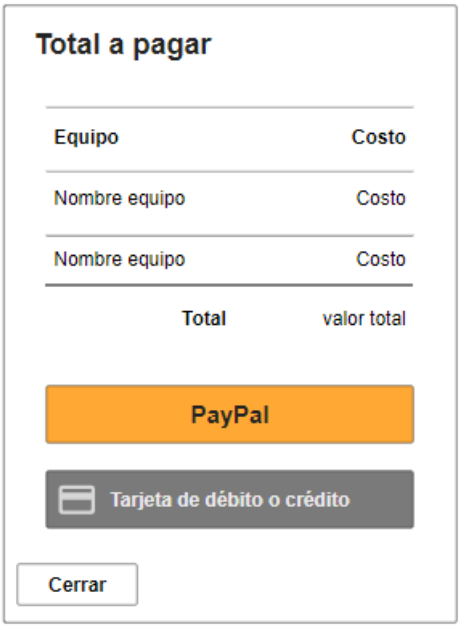


Figura 3.13: Interfaz metódo de pago
Fuente: Elaborado por el investigador

En la pantalla de suscripciones se listan los distintos planes con sus descripciones y precios a los que el representante del equipo puede suscribirse

Figura 3.14: Interfaz planes de suscripción
Fuente: Elaborado por el investigador

En la pantalla video se reproduce el video que el representante de equipo haya seleccionado, la información del usuario propietario del video y un botón para enviar mensaje y contactarse con el usuario.

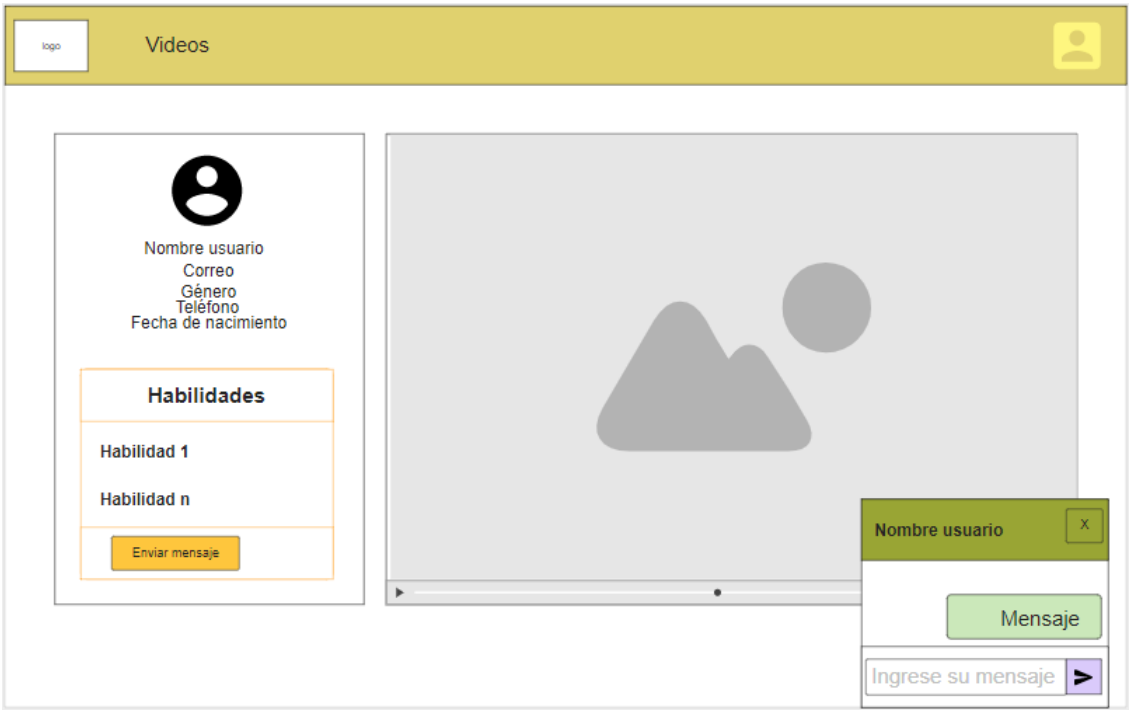
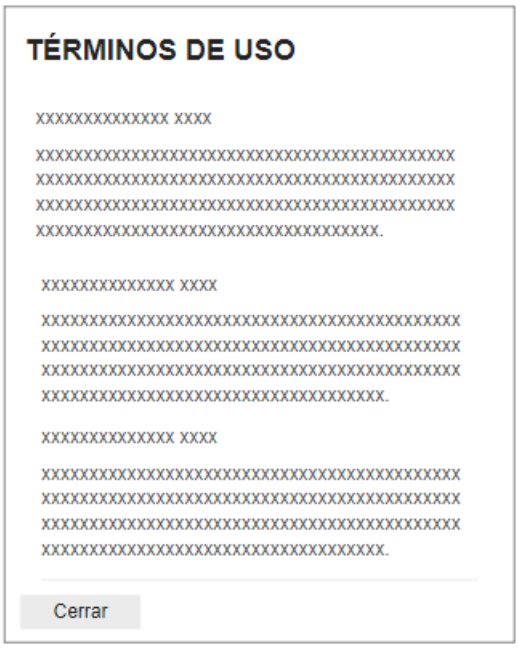


Figura 3.15: Interfaz reproducción de video
Fuente: Elaborado por el investigador

La ventana modal de términos de uso describe los términos y condiciones de uso

del sistema.



TÉRMINOS DE USO

xxxxxxxxxxxxxx xxxx

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

xxxxxxxxxxxxxx xxxx

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

xxxxxxxxxxxxxx xxxx

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

Figura 3.16: Interfaz términos y condiciones de uso
Fuente: Elaborado por el investigador

3.2.3. Fase 3: Codificación

3.2.3.1. Estructura del servidor (Back-end)

Para desarrollar el api rest se utilizó node js con express; se definieron distintas carpetas como son app, node_modules, imágenes y videos.

En la carpeta app se definieron los archivos de configuración, los métodos con las distintas funcionalidades del api y las rutas organizados por carpetas, de la siguiente manera:

Configuración: esta carpeta contiene dos archivos de configuración uno para la clave secreta utilizada para el token de sesión y otro para la configuración de la base de datos con la que se va a trabajar.

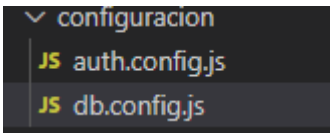


Figura 3.17: Carpeta configuración
Fuente: Elaborado por el investigador

Controladores: como su nombre lo indica, esta carpeta contiene los archivos

controladores donde se desarrollan cada uno de los métodos con las funcionalidades requeridas del api.

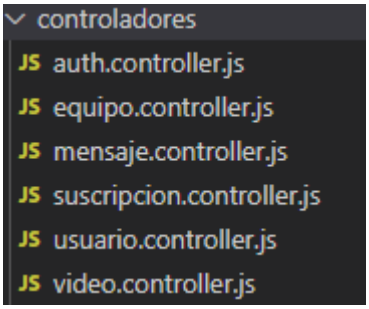


Figura 3.18: Carpeta controladores
Fuente: Elaborado por el investigador

Middleware: en esta carpeta se encuentran los archivos que sirven para realizar verificaciones y controles respectos al inicio de sesión, tipo de usuario, email duplicado y existencia de un rol.

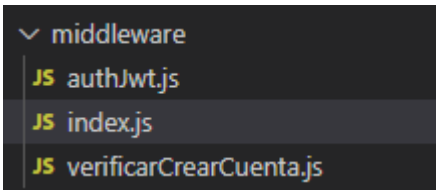


Figura 3.19: Carpeta middleware
Fuente: Elaborado por el investigador

Modelos: en esta carpeta se definen todos los modelos de las tablas de la base de datos, especificando su nombre y sus campos con el tipo de dato.

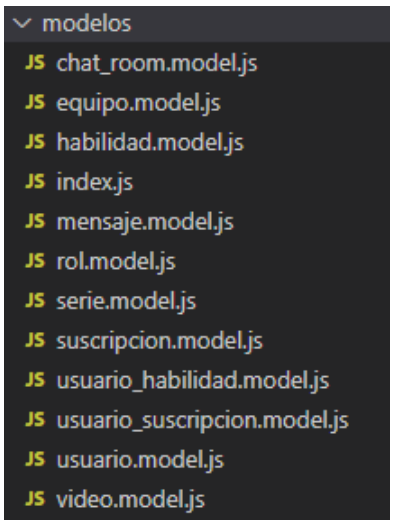


Figura 3.20: Carpeta modelos
Fuente: Elaborado por el investigador

Rutas: esta carpeta posee todas las rutas con las que se va a trabajar para el manejo de los datos organizado en distintas carpetas.

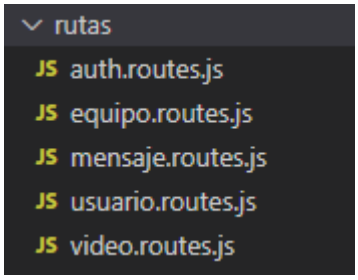


Figura 3.21: Carpeta rutas
Fuente: Elaborado por el investigador

En la carpeta de node_modules se importan todas las librerías que se han instalado dentro del proyecto.
La carpeta de imágenes y videos sirven para guardar los videos que los usuarios suben y la captura de las miniaturas de los mismos.

3.2.3.2. Instalación de librerías

Dentro del área del cliente se implementaron las librerías de ng-2-fileuplaod para la subida de videos
Por la parte del servidor se implementó sequelize para el manejo de la base de datos, body-parser, mysql, express, con el siguiente comando:

Figura 3.22: Instalación de módulos
Fuente: Elaborado por el investigador

Para el manejo de las imágenes y videos se utilizó multer, ffprobe y fluent-ffmpeg, mientras que para el desarrollo de las tareas programadas se implementó node-cron; para la instalación de todas las dependencias se utilizaron los comandos de la figura 3.23.

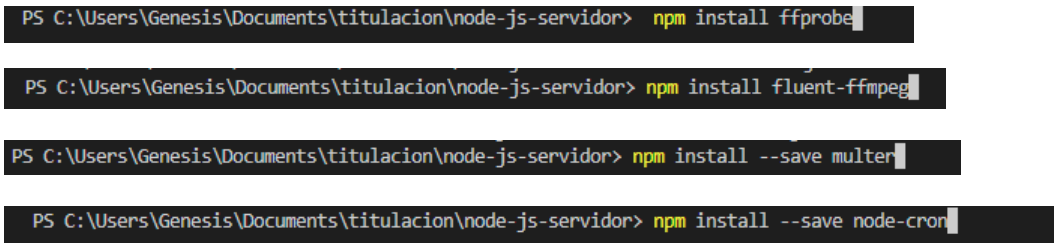


Figura 3.23: Instalación de módulos
Fuente: Elaborado por el investigador

3.2.3.3. Configuración de la base de datos

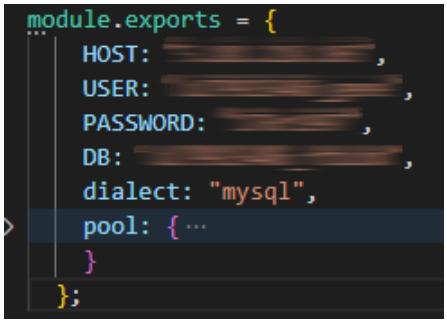


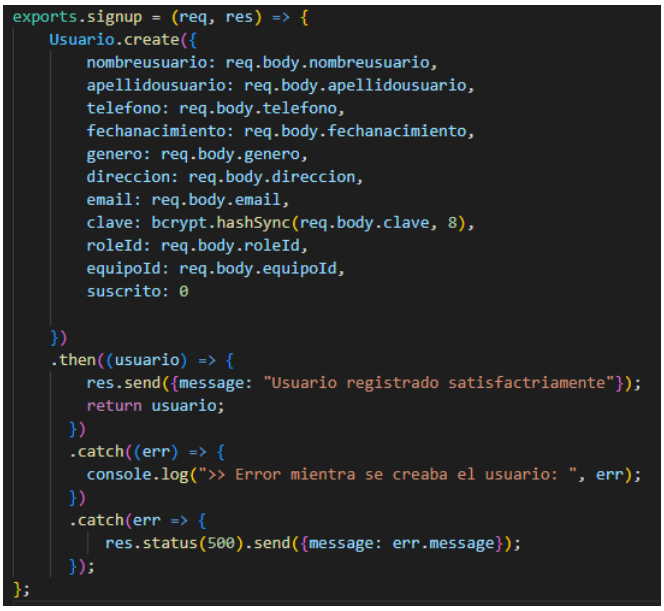
Figura 3.24: Configuración de la base de datos
Fuente: Elaborado por el investigador

Host: URL o dirección IP de servidor donde se aloja la base de datos.
User: nombre de usuario con el que accedemos a la base de datos
Password: clave de acceso a la base de datos
DB: nombre de la base de datos
dialec: definimos la base de datos que estamos utilizando.
pool: grupo de conexiones

3.2.3.4. Métodos de creación de cuenta e inicio de sesión

En los métodos de creación de cuenta e inicio de sesión tenemos el registro de usuario, email duplicado, existencia del rol asignado, inicio de sesión y verificar token de sesión.

El método presentado en la figura 3.25 permite la creación de un nuevo usuario, para lo cual requiere su información general (nombre, apellido, teléfono, fecha de nacimiento, dirección, género, email, clave y rol), en caso de ser usuario directivo también es necesario el id del equipo al que representa.



```
exports.signup = (req, res) => {
    Usuario.create({
        nombreusuario: req.body.nombreusuario,
        apellidousuario: req.body.apellidousuario,
        telefono: req.body.telefono,
        fechanacimiento: req.body.fechanacimiento,
        genero: req.body.genero,
        direccion: req.body.direccion,
        email: req.body.email,
        clave: bcrypt.hashSync(req.body.clave, 8),
        roleId: req.body.roleId,
        equipoId: req.body.equipoId,
        suscrito: 0

    })
    .then((usuario) => {
        res.send({message: "Usuario registrado satisfactriamente"});
        return usuario;
      })
      .catch((err) => {
        console.log(">> Error mientra se creaba el usuario: ", err);
      })
      .catch(err => {
          res.status(500).send({message: err.message});
      });
};
```

Figura 3.25: Registro de usuario
Fuente: Elaborado por el investigador

El siguiente método permite verificar si el email que se está ingresando ya existe con el fin de controlar que los correos de los usuarios sean únicos.

```
verificarEmailDuplicado = (req, res, next) => {
    Usuario.findOne({
        where : {
            email: req.body.email
        }
    }).then (usuario => {
        if (usuario) {
            res.status(400).send({
                message: "Error. El correo ya esta en uso"
            });
            return;
        }
        next ();
    });
};
```

Figura 3.26: Email duplicado
Fuente: Elaborado por el investigador

El siguiente método sirve para controlar que el rol que se está asignando a un usuario exista dentro de la base de datos.

```
verificarSiExisteRol = (req, res, next) => {
    if (req.body.roleId) {
          if (!ROLES.includes(req.body.roleId)) {
              res.status(400).send({
                  message:"Error. El rol no existe = " + req.body.roles[i]
              });
              return;
          }
    }
    next();
};
```

Figura 3.27: Rol existente
Fuente: Elaborado por el investigador

El método singin sirve para el inicio de sesión, para este se requiere el correo y la clave que el usuario digitó al crear su cuenta, con esta información se valida que exista el usuario y la clave sea correcta.

Figura 3.28: Inicio de sesión
Fuente: Elaborado por el investigador

El método verificarToken permite controlar el inicio de sesión del usuario.

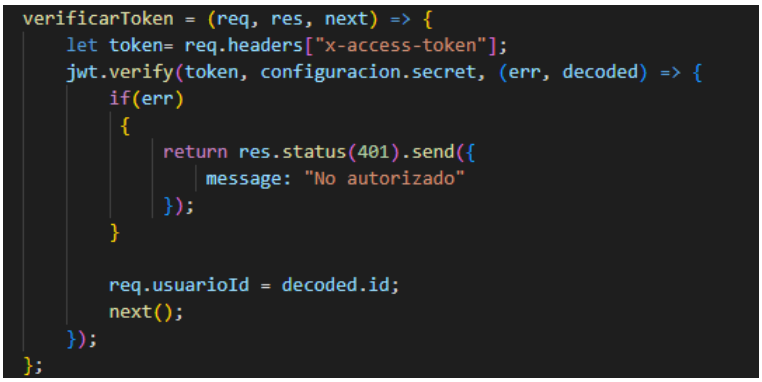


Figura 3.29: Verificar token
Fuente: Elaborado por el investigador

3.2.3.5. Métodos del usuario

Dentro de los métodos del usuario están los métodos para editar usuario, actualizar el estado del usuario, actualizar el estado de la suscripción del usuario, listar habilidades, guardar, listar y contar las habilidades del usuario.
El método actualizar permite editar la información del usuario, esta puede ser nombre, apellido, dirección, clave, teléfono, género y fecha de nacimiento.

Figura 3.30: Actualizar usuario
Fuente: Elaborado por el investigador

El método actualizar estado sirve para activar la cuenta del usuario, cambiando su estado activo de 0 a 1.

Figura 3.31: Actualizar estado de usuario
Fuente: Elaborado por el investigador

El método actualizar estado suscripción permite cambiar el estado del campo suscrito del usuario directivo de 0 a 1 o viceversa, según sea el caso.

```
exports.actualizarEstadoSuscripcion= (req, res) => {
  const id = req.params.id;
  if(!req.body.clave){

      Usuario.update(
          {
              suscrito: req.body.suscrito
          },
          {where: { id:id}}
      )
      .then((num) => {
          if(num > 0) {
              res.send({
                  message: "Usuario actualizado correctamente",
              });
          }else {
              res.send({
                  message: "No se puede actualizar el usuario",
              });
          }
         })
          .catch((err) => {
          console.log(">> Error mientras se actualizaba el usuario: ", err);
          res.status(500).send({message: err.message});
      });
  }

}
```

Figura 3.32: Actualizar estado de la suscripción del usuario
Fuente: Elaborado por el investigador

El método de listar habilidades permite obtener todas las habilidades que un jugador puede tener.

```
exports.listarHabilidades = (req, res) => {
    Habilidad.findAll()
    .then(data => {

      res.send(data);
    })
    .catch(err => {
      res.status(500).send({
        message:
        err.message || "Error al encontrar los datos"
      });
    });
  };
```

Figura 3.33: Listar habilidades
Fuente: Elaborado por el investigador

El método guardar sirve para enviar a la base de datos las habilidades de un usuario.

```
exports.guardar =(req, res) => {
  console.log(req.body.habilidades);

  var data = [];
  let i=0;
  req.body.habilidades.forEach(element => {
    data[i] = {
      'usuarioId': req.body.usuarioId,
      'habilidadId': element
    }
    i++;
  });
  UsuarioHabilidad.bulkCreate(data, {individualHooks: true}).then((usuariohabilidad) => {
    res.send({message: "Habilidades registradas satisfactriamente"});
  })
  .catch((err) => {
    console.log(">> Error mientras se guardaban las habilidades: ", err);
  });
}
```

Figura 3.34: Guardar habilidades del usuario
Fuente: Elaborado por el investigador

El siguiente método permite contar todas las habilidades que el usuario posee, con el fin de verificar si tiene o no habilidades guardadas.

```
exports.contarHabilidades = (req, res) => {
  const idUsuario = req.params.id;
  sequelize.query(`SELECT  COUNT(h.nombre) as total
                   FROM usuario_habilidads uh JOIN usuarios u
                   ON uh.usuarioId= u.id
                   JOIN habilidads h
                   ON  h.id = uh.habilidadId
                   AND u.id =
                ` + idUsuario, {
    type: Sequelize.QueryTypes.SELECT,
  })
  .then((data) => {
    res.send(data);
  })
  .catch((err) => {
    res.status(500).send({
      message:
        err.message ||
        "Error al contar las habilidades del usuario",
    });
  });
};
```

Figura 3.35: Contar habilidades del usuario
Fuente: Elaborado por el investigador

El método listar habilidades por usuario permite obtener todas las habilidades que un usuario específico posee.

```
exports.listarHabilidadesPorUsuario = (req, res) => {
    const idUsuario = req.params.id;
    sequelize.query(`SELECT  h.nombre
                     FROM usuario_habilidads uh JOIN usuarios u
                     ON uh.usuarioId = u.id
                     JOIN habilidads h
                     ON  h.id = uh.habilidadId
                     AND u.id =
                ` + idUsuario, {
      type: Sequelize.QueryTypes.SELECT,
    })
    .then((data) => {
      res.send(data);
    })
    .catch((err) => {
      res.status(500).send({
        message:
          err.message ||
          "Error al obtener las habilidades del usuario",
      });
    });
};
```

Figura 3.36: Listar habilidades del usuario
Fuente: Elaborado por el investigador

3.2.3.6. Métodos del equipo

En los métodos del equipo tenemos listar series, crear equipos y listar equipos. Con el siguiente método se puede obtener todas las series existentes en el sistema.

```
exports.listarSeries = (req, res) => {
    Serie.findAll()
    .then(data => {
      res.send(data);
    })
    .catch(err => {
      res.status(500).send({
        message:
          err.message || "Error al encontrar los datos"
        });

    });
};
```

Figura 3.37: Listar series
Fuente: Elaborado por el investigador

Con el método crear equipos se puede guardar información (nombre, dirección, teléfono y serie) de un nuevo equipo.

```
exports.crearEquipo = (req, res) => {
    Equipo.create({
        nombre: req.body.nombre,
        direccion: req.body.direccion,
        telefono: req.body.telefono,
        serieId: req.body.serieId
    })
    .then((equipo) => {
        res.send({message: "Usuario registrado satisfactriamente"});
        return equipo;
      })
      .catch((err) => {
        console.log(">> Error mientras se creaba el usuario: ", err);
      })
      .catch(err => {
          res.status(500).send({message: err.message});
      });
};
```

Figura 3.38: Crear equipo
Fuente: Elaborado por el investigador

El método de listar equipos permite obtener todos los equipos existentes en el sistema.

```
exports.listarEquipos = (req, res) => {
  Equipo.findAll()
  .then(data => {
    res.send(data);
  })
  .catch(err => {
    res.status(500).send({
      message:
      err.message || "Error al encontrar los datos"
    });
  });
}
```

Figura 3.39: Listar equipos
Fuente: Elaborado por el investigador

3.2.3.7. Métodos de mensajes

Los métodos de mensajes son: listar mensajes, listar chat rooms, guardar mensaje, actualizar estado del mensaje y contar mensajes nuevos.
El método de listar mensajes permite listar los mensajes de un chat-room específico.

```
exports.listarMensajes = (req, res) => {
        const idChat = req.params.id;
        Mensaje.findAll(
          {where: { chatRoomId: idChat}}
        )
        .then(data => {

          res.send(data);
        })
        .catch(err => {
          res.status(500).send({
            message:
            err.message || "Error al encontrar los datos"
          });
        });
};
```

Figura 3.40: Listar mensajes
Fuente: Elaborado por el investigador

Con el método listar chat-rooms se puede obtener todos los chat-rooms pertenecientes a un usuario.

```
exports.listarChatRooms = (req, res) => {
  const idUsuario = req.params.id;
  sequelize.query(`SELECT c.id, c.usuarioUId, u.nombreusuario nombreusuarioU, u.apellidousuario apellidoUsuarioU, c.usuarioDId, ud.nombreusuario
                   (SELECT texto FROM mensajes where id = (SELECT MAX(id) FROM mensajes WHERE chatRoomId = c.id)) mensaje,
                   (SELECT usuarioId FROM mensajes where id = (SELECT MAX(id) FROM mensajes WHERE chatRoomId = c.id)) emisor,
                   (SELECT leido FROM mensajes where id = (SELECT MAX(id) FROM mensajes WHERE chatRoomId = c.id)) estado
                   FROM chat_rooms c JOIN usuarios u
                   ON c.usuarioUId = u.id
                   JOIN usuarios ud
                   ON ud.id = c.usuarioDId
                   WHERE c.usuarioUId = `+ idUsuario +
                   ` OR c.usuarioDId=
            ` + idUsuario , {
    type: Sequelize.QueryTypes.SELECT,
  })
  .then((data) => {
    res.send(data);
  })
  .catch((err) => {
    res.status(500).send({
      message:
        err.message ||
        "Error al obtener los mensajes.",
    });
  });
};
```

Figura 3.41: Listar chat rooms
Fuente: Elaborado por el investigador

El método guardar mensaje permite que un usuario se pueda contactar con otro por medio de mensajes dentro del sitio, primero verifica que si existe un chat-room con los id del usuario emisor y receptor para almacenar la información del mensaje (mensaje, id del chat-room, id del emisor y estado del mensaje); en caso de no existir el chat-room con estos id el método crea un nuevo chat-room y almacena la información del mensaje.

```
exports.guardadMensaje = (req, res) => {
  if (!req.body.usuarioEmisor || !req.body.usuarioReceptor || !req.body.mensaje ) {
    res.status(400).send({
      message: "Debe ingresar todos los campos",
    });
    return;
  }
  ChatRoom.findOne({
    where : {

      [Op.or]: [
        { usuarioUId: req.body.usuarioEmisor,
          usuarioDId: req.body.usuarioReceptor },
        { usuarioUId: req.body.usuarioReceptor,
          usuarioDId: req.body.usuarioEmisor }
      ]

    }
}).then (chat => {
  console.log(chat)
    if (chat) {
      Mensaje.create({
        usuarioId: req.body.usuarioEmisor,
        chatRoomId: chat.id,
        texto: req.body.mensaje,
        leido: 0
      })
      .then((mensaje) => {
        res.send({message: "Mensaje enviado satisfactriamente"});
        return mensaje;
      })
        .catch(err => {
            res.status(500).send({message: err.message});
        });
      }else{

    // Guardamos factura y su detalle
    ChatRoom.create({
      usuarioUId: req.body.usuarioEmisor,
      usuarioDId: req.body.usuarioReceptor
    })
      .then((data) => {
        console.log(data.id);
        if (req.body.mensaje) {

          Mensaje.create({
            chatRoomId: data.id,
            usuarioId: req.body.usuarioEmisor,
            texto: req.body.mensaje,
            leido: 0
          })
            .then(() => {
              res.send({
                message: "Mensaje enviado correctamente",
              });
            })
            .catch((err) => {
              res.status(500).send({ message: err.message });
            });
        } else {
          res.status(404).send({
            message: "No se puede enviar un mensaje vacio",
          });
        }
```

Figura 3.42: Guardar mensaje
Fuente: Elaborado por el investigador

El método contar mensajes nuevos permite saber si existen mensajes nuevo o no para un usuario específico; este método cuenta todos los mensajes de un usuario siempre y cuando el campo leído sea igual a cero.

```
exports.contarMensajesNuevos = (req, res) => {
  const idUsuario = req.params.id;
  sequelize.query(`SELECT COUNT(m.id) total
                   FROM mensajes m JOIN chat_rooms c ON
                   m.chatRoomId = c.id
                   WHERE ( c.usuarioUId = `+ idUsuario +
                   ` OR c.usuarioDId = ` + idUsuario + `)
                   AND m.leido = 0
                   AND NOT m.usuarioId = ` + idUsuario , {
       type: Sequelize.QueryTypes.SELECT,
    })
    .then((data) => {
    res.send(data);
    })
    .catch((err) => {
    res.status(500).send({
    message:
    err.message ||
    "Error al obtener los mensajes.",
    });
    });
}
```

Figura 3.43: Contar mensajes
Fuente: Elaborado por el investigador

El método de actualizar estado de mensaje permite cambiar es estado de leído de 0 a 1 una vez que el usuario receptor haya abierto dicho mensaje.

```
exports.actualizarEstadoMensaje = (req, res) => {

 Mensaje.update({
      leido: 1,
   },
   {
     where: {
      chatRoomId: req.body.id,
      [Op.not]: [
        {usuarioId: req.body.usuarioEmisor},
      ]
     }
   }
   )
   .then((mensaje) => {
      res.send({message: "Datos actualizados exitosamente"});
      return mensaje;
    })
    .catch((err) => {
      console.log(">> Error mientras se actualizaban los datos: ", err);
    })
    .catch(err => {
        res.status(500).send({message: err.message});
    });
}
```

Figura 3.44: Actualizar estado del mensaje
Fuente: Elaborado por el investigador

3.2.3.8. Métodos de suscripción

Los métodos de suscripción son: listar suscripciones y guardar la suscripción seleccionada por el representante del equipo.

El método listar suscripciones obtiene la información (nombre, descripción y precio) de todos los tipos de suscripciones existentes en el sistema

```
exports.listarSuscripciones = (req, res) => {
  Suscripcion.findAll()
  .then(data => {

    res.send(data);
  })
  .catch(err => {
    res.status(500).send({
      message:
      err.message || "Error al encontrar los datos"
    });
  });
};
```

Figura 3.45: Listar suscripciones
Fuente: Elaborado por el investigador

El método guardar almacena el id de la suscripción seleccionada, la fecha de inicio de la suscripción, fecha de fin y el id del usuario que se ha suscrito dentro de la base de datos.

```
exports.guardar = (req, res) => {

  UsuarioSuscripcion.create({
      usuarioId: req.body.usuarioId,
      suscripcionId: req.body.suscripcionId,
      fechaInicio: req.body.fechaInicio,
      fechaFin: req.body.fechaFin
  })
  .then((usuario_suscripcion) => {
      res.send({message: "Suscripcion guardada satisfactriamente"});
      return usuario_suscripcion;
    })
    .catch((err) => {
      console.log(">> Error mientras se guardaba la suscripcion: ", err);
    })
    .catch(err => {
        res.status(500).send({message: err.message});
    });
};
```

Figura 3.46: Guardar suscripción
Fuente: Elaborado por el investigador

3.2.3.9. Métodos del video

Entre los métodos de video están los siguientes: guardar video, reproducir video, listar videos, listar videos no vistos y vistos, buscar videos, actualizar estado de video.

El siguiente método permite almacenar las URLs del video y su miniatura, el título, descripción y id del usuario propietario del video.

```
exports.guardar = (req, res) => {
    Video.create({
        url:req.body.url,
        imagen: req.body.imagen,
        titulo: req.body.titulo,
        descripcion: req.body.descripcion,
        usuarioId: req.body.usuarioId,
    })
    .then(video => {
        if(req.body.equipos){
          Equipo.findAll ({
            where: {
              id: {
                [Op.or]: req.body.equipos
              }
            }
          }).then(equipos => {
            video.setEquipos(equipos).then(()=> {
              res.send({message: "Video guardado exitosamente"});
            });
          });
        }else {
          video.setEquipos([1]).then(()=> {
            res.send({message: "Video guaradado exitosamente"});
          });
        }
    })
    .catch(err => {
      res.status(500).send({message: err.message});
    });
};
```

Figura 3.47: Guardar video
Fuente: Elaborado por el investigador

El siguiente método permite reproducir un video en específico del servidor en el cliente.

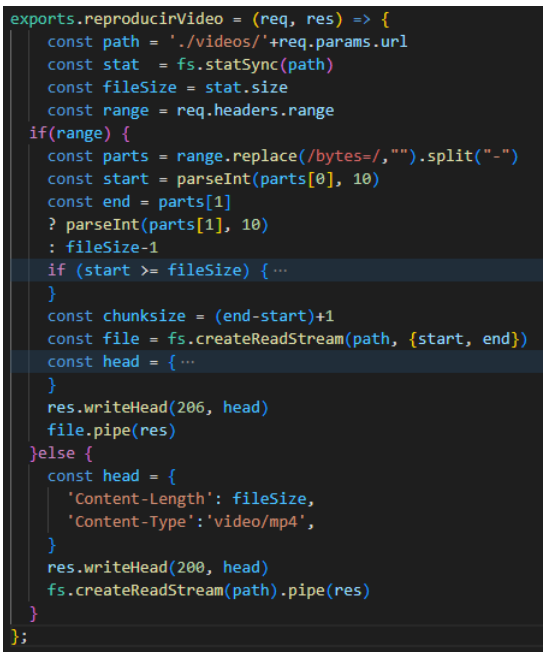


Figura 3.48: Reproducir video
Fuente: Elaborado por el investigador

El método listar videos obtiene todos los video que han sido subidos para un equipo.

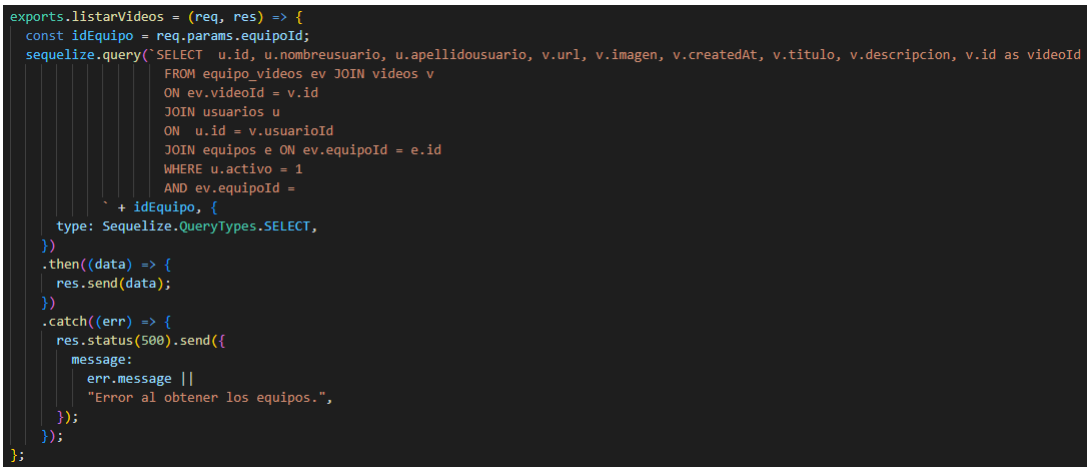


Figura 3.49: Listar videos
Fuente: Elaborado por el investigador

El método buscar video permite la búsqueda de videos según el título, nombre de usuario, apellido de usuario o descripción.

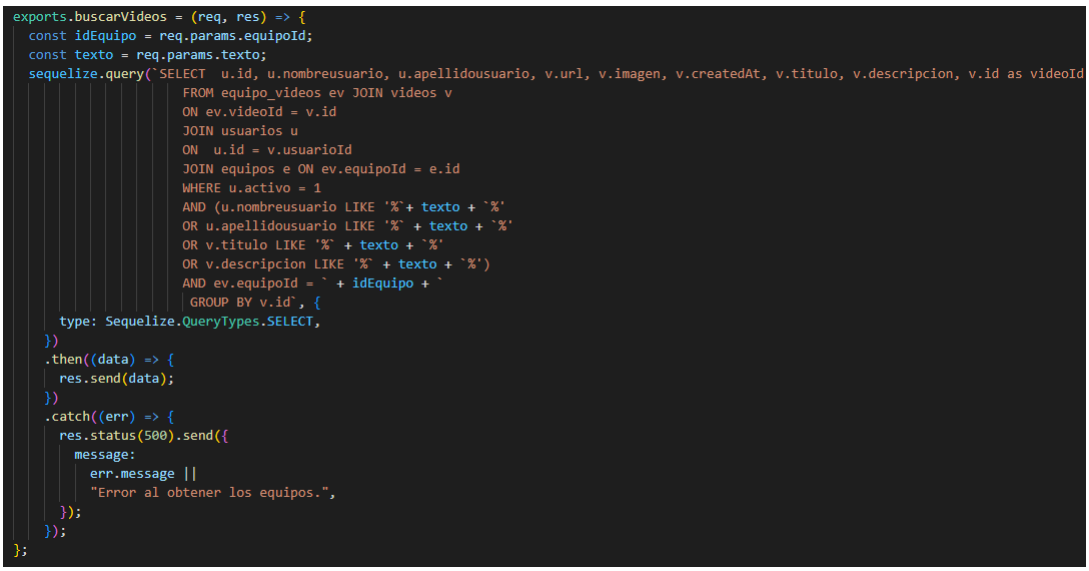


Figura 3.50: Buscar video
Fuente: Elaborado por el investigador

El método listar videos no vistos obtiene todos los videos nuevo para un equipo específico.

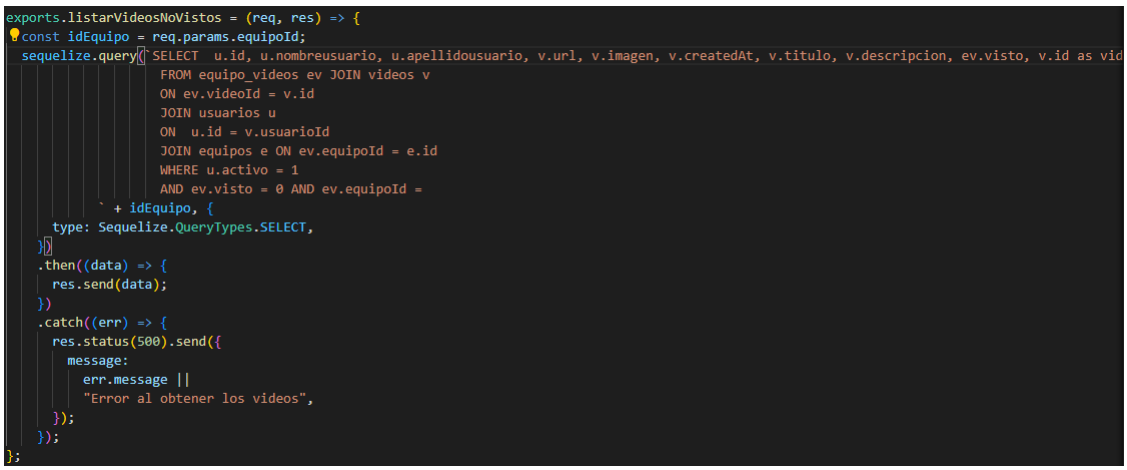


Figura 3.51: Listar videos no vistos
Fuente: Elaborado por el investigador

El método listar videos vistos obtiene todos los videos que ya hayan sido reproducidos por el usuario directivo de un equipo específico.

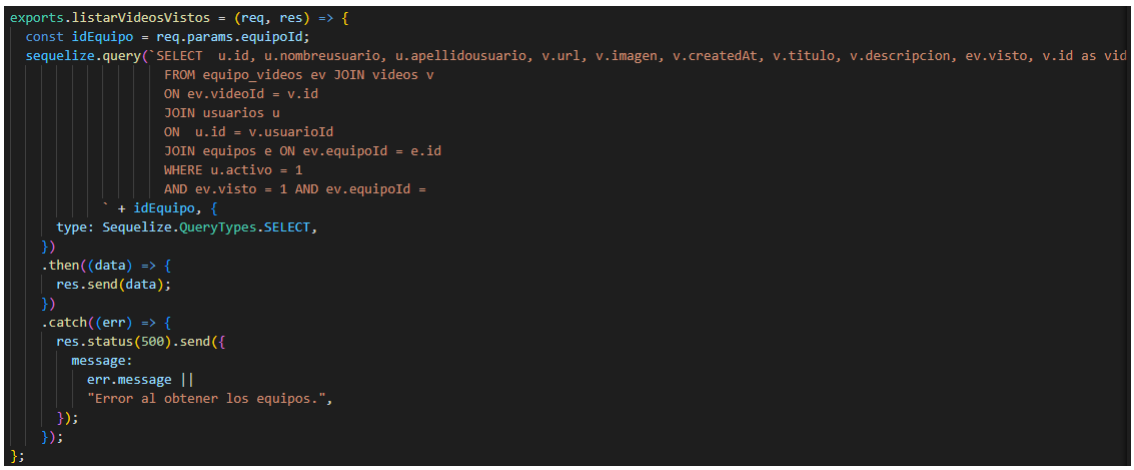


Figura 3.52: Listar videos vistos
Fuente: Elaborado por el investigador

El método actualizar estado de video permite cambiar el estado de no visto (nuevo) a visto, cambiando el campo leído de 0 a 1.

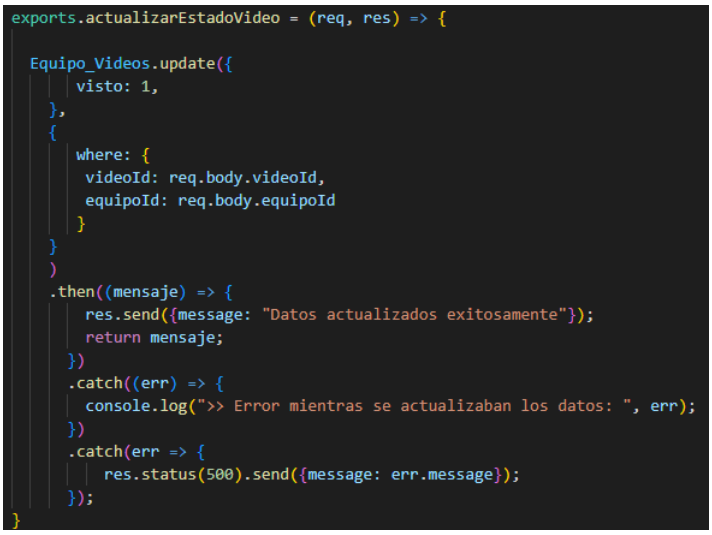


Figura 3.53: Actualizar estado de video
Fuente: Elaborado por el investigador

3.2.3.10. Tareas programadas

Las tareas programadas que el sistema tienen sirven para actualizar el estado de la suscripción del representante de equipo y el estado del usuario.
La tarea programada para actualizar el estado suscrito del representante de equipo se ejecuta todos los días por la noche; esta tarea verifica que la fecha final de la suscripción sea igual a la actual, si es así el estado de suscrito del representante de equipo cambiara a 0 y no podrá reproducir videos hasta que se vuelva a suscribir.

Figura 3.54: Tarea programada para actualizar el estado de suscripción
Fuente: Elaborado por el investigador

La tarea programada para actualizar el estado del usuario se ejecuta el 28 de cada mes por la noche y sirve para editar el estado del usuario de activo a inactivo en caso de no haber subido ningún video durante un mes.

```
cron.schedule('0 0 28 * *', function(){

controler.actualizarEstadoUsuario();

console.log("Tarea actualizandose")


})
```

```
exports.actualizarEstadoUsuario = (req, res) => {
  var date = new Date();
  var local_fecha = date.toLocaleDateString();
  const fecha = local_fecha.split('/');
  var mes = fecha[1] -1;
  const fechaActual = fecha[2] +'-'+ fecha[1] + '-' + fecha[0];
  const fechaInicial = fecha[2]+'-'+ mes +'-' + '01';
  console.log(mes);
sequelize.query(`UPDATE usuarios u SET u.activo = 0
                 WHERE (SELECT count(*)
                                FROM videos v
                                WHERE CAST(createdAt AS DATE) BETWEEN ` + fechaInicial + ` AND ` + fechaActual +
                                ` AND v.usuarioId = u.id) < 1
                 AND u.roleId = 3` , {
      type: Sequelize.QueryTypes.UPDATE,
      })
      .then((data) => {
        console.log(data);
      })
      .catch((err) => {

      console.log("Error al obtener las habilidades del usuario", err.message);
      });
}
```

Figura 3.55: Tarea programada para actualizar el estado del usuario
Fuente: Elaborado por el investigador

3.2.3.11. Integración de Paypal en el cliente

Para integrar el botón de Paypal como método de pago primero se crea una cuenta developed.

Una vez creada la cuenta developed e iniciar sesión, dar clic en la opción Apps & Credentials;

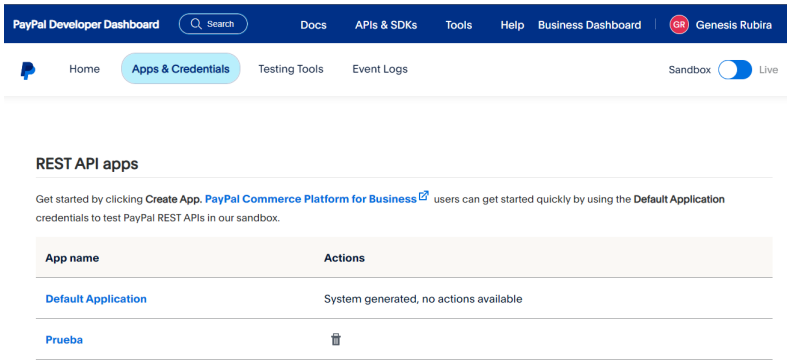


Figura 3.56: Ventana Apps & Credentials de PayPal
Fuente: Elaborado por el investigador

en esta ventana se puede crear una nueva aplicación o utilizar la que viene por defecto. De la aplicación creada se copia el Client ID para reemplazarla en el

script que ha sido añadido en el archivo index.html del proyecto.

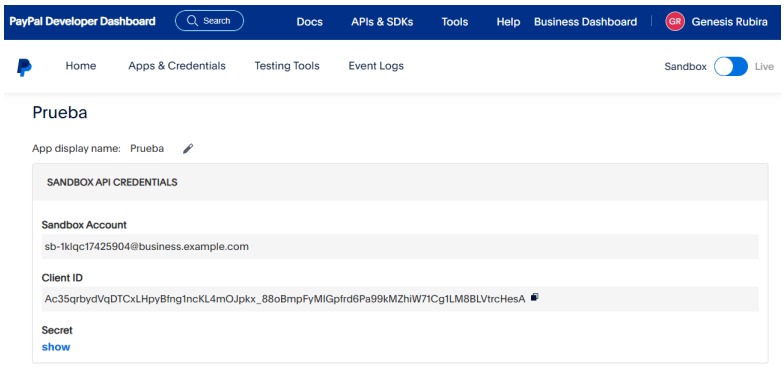


Figura 3.57: Ventana de información del app
Fuente: Elaborado por el investigador

```
<script
src="https://www.paypal.com/sdk/js?client-id=Ac35qrbydVqDTCxLHpyBfng1ncKL4mOJpkx_88oBmpFyMIGpfrd6Pa99kMZhiW71Cg1LM8BLVtrcHesA&currency=USD">
</script>
```

Figura 3.58: Script para la integración de Paypal
Fuente: Elaborado por el investigador

En este script se configura el client id de paypal y el tipo de moneda con la que se trabajará.

3.2.3.12. Consumo del api desde el cliente

Para consumir los métodos del servidor se utilizan las rutas creadas en el api.

1. **Métodos de registro e inicio de sesión.**
   Para consumir los métodos de registro e inicio de sesión primero se configura la ruta con la que se accederá a cada uno de ellos, así como se puede ver en la figura 3.59.

```
const AUTH_API = 'https://node-js-servidor-production.up.railway.app/api/auth/';
```

Figura 3.59: Ruta del api para los métodos de registro e inicio de sesión
Fuente: Elaborado por el investigador

   El método registrar permite enviar la información del usuario (nombre, apellido, teléfono, fecha de nacimiento, género, dirección, email y clave) desde el cliente al servidor para crear la cuenta del usuario.

```
registar(user: any): Observable<any> {
  return this.http.post(AUTH_API + 'crearcuenta', {
      nombreusuario: user.nombreusuario,
      apellidousuario: user.apellidousuario,
      telefono: user.telefono,
      fechanacimiento: user.fechanacimiento,
      genero: user.genero,
      direccion: user.direccion,
      email: user.email,
      clave: user.clave,
      roleId: 3
  }, httpOptions);
```

Figura 3.60: Consumo del método para registrarse
Fuente: Elaborado por el investigador

El método iniciar sesión permite que el usuario acceda al sitio web por medio de su correo y clave.

```
iniciarsesion(credentials: any): Observable<any> {
  return this.http.post(AUTH_API + 'iniciarsesion', {
    email: credentials.email,
    clave: credentials.clave
  }, httpOptions);
}
```

Figura 3.61: Consumo del método para iniciar sesión
Fuente: Elaborado por el investigador

2. **Métodos del usuario.**
   La figura 3.62 es la ruta por la que se accede a los métodos del usuario.

```
const API_URL = 'https://node-js-servidor-production.up.railway.app/api/test/';
```

Figura 3.62: Ruta del api para los métodos del usuario
Fuente: Elaborado por el investigador

El método editar usuario permite enviar los datos que el usuario va a editar de su información personal.

```
editarUsuario(id:number, user:any): Observable<any> {
  return this.http.put(API_URL + 'editar/'+ id,
  {
    nombreusuario:  user.nombreusuario,
    apellidousuario: user.apellidousuario,
    telefono: user.telefono,
    direccion: user.direccion,
    clave: user.clave

  }, httpOptions);
}
```

Figura 3.63: Consumo del método para editar usuario
Fuente: Elaborado por el investigador

El método de editar estado enviá el id del usuario del cual se va a editar el estado para que el servidor ejecute el método de actualizar el campo activo del usuario.

```
editarEstado(id:number): Observable<any> {
  return this.http.put(API_URL + 'Actualizar/'+ id, httpOptions);
}
```

Figura 3.64: Consumo del método para editar estado del usuario
Fuente: Elaborado por el investigador

El método editar suscripción cambia es estado de suscripción del usuario cambiando el campo suscrito de 0 a 1 según el caso.

```
aditarSuscripcion(id: number, estado: any): Observable<any> {
  return this.http.put(API_URL + 'editarEstadoSuscripcion/'+ id, {
    suscrito: estado
  }, httpOptions);
}
```

Figura 3.65: Consumo del método para editar estado de la suscripción del representante de equipo
Fuente: Elaborado por el investigador

El método obtener usuario por id permite obtener los datos de un usuario específico.

```
obtenerUsuarioPorId (id:number):Observable<any> {
  return this.http.get(API_URL + 'obtener/'+ id, {responseType: 'text'});
}
```

Figura 3.66: Consumo del método para obtener el usuario por el id
Fuente: Elaborado por el investigador

El método obtener habilidades permite listar todas las habilidades que un usuario puede tener por medio del método obtenerHabilidades del api.

```
obtenerHablidades():Observable<any> {
  return this.http.get(API_URL + 'obtenerHabilidades',{responseType: 'text'});
}
```

Figura 3.67: Consumo del método para listar habilidades
Fuente: Elaborado por el investigador

El método obtener habilidades por usuario sirve para llamar obtener las habilidades de un usuario.

```
obtenerHabilidadesPorUsuario(id: number): Observable<any> {
  return this.http.get(API_URL + 'obtenerHabilidadesUsuario/' + id, {responseType: 'text'});
}
```

Figura 3.68: Consumo del método para listar las habilidades del usuario
Fuente: Elaborado por el investigador

El siguiente método permite guardar todas las habilidades que un usuario tiene.

```
guardarHabilidades(usuario:any, habilidades:any): Observable<any> {

  return this.http.post(API_URL+'guardarHabilidades', {
    usuarioId:usuario,
    habilidades: habilidades
  }, httpOptions);
};
```

Figura 3.69: Consumo del método para guardar las habilidades del usuario
Fuente: Elaborado por el investigador

El método de contar habilidades permite verificar si el usuario posee o no alguna habilidad llamando al método del api para contar habilidades del usuario.

```
contarHabilidades(id: number): Observable<any> {
  return this.http.get(API_URL + 'contarHabilidadesUsuario/' + id, {responseType: 'text'});
}
```

Figura 3.70: Consumo del método para contar las habilidades del usuario
Fuente: Elaborado por el investigador

3. **Métodos del equipo.**

   Ruta para acceder a los métodos del equipo

```
const API_URL = 'https://node-js-servidor-production.up.railway.app/api/equipo/';
```

Figura 3.71: Ruta del api para los métodos del equipo
Fuente: Elaborado por el investigador

El método crear equipo sirve para enviar la información del nuevo equipo (nombre, dirección, teléfono e id de la serie) para guardarlo.

```
crearEquipo(equipo: any, serieId: any): Observable<any> {

  return this.http.post(API_URL + 'crear', {
      nombre: equipo.nombre,
      direccion: equipo.direccion,
      telefono: equipo.telefono,
      serieId: serieId
  }, httpOptions);
};
```

Figura 3.72: Consumo del método para crear un equipo
Fuente: Elaborado por el investigador

El método obtener series permite listar todas las series existentes en el sistema.

```
obtenerSeries(): Observable<any> {
const listado= this.http.get('https://node-js-servidor-production.up.railway.app/api/serie/listar', {responseType: 'text'});
return listado
}
```

Figura 3.73: Consumo del método para listar series
Fuente: Elaborado por el investigador

El método obtener equipos permite listar todos los equipos que existen en el sistema.

```
obtenerEquipos(): Observable<any> {
  return this.http.get(API_URL + 'listar', {responseType: 'text'});
}
```

Figura 3.74: Consumo del método para listar equipos
Fuente: Elaborado por el investigador

4. **Métodos de mensajes**

   El siguiente método obtener todos los mensajes que han sido enviados para un usuario en específico.

```
obtenerMensajes(id: number): Observable<any> {
  return this.http.get('https://node-js-servidor-production.up.railway.app/api/obtenerMensajes/' + id, {responseType: 'text'});
}
```

Figura 3.75: Método para consumir el método de listar mensajes
Fuente: Elaborado por el investigador

El método obtener chat-rooms permite listar todos los chat-rooms que un usuario tiene.

```
obtenerChatRooms(id: number): Observable<any> {
  return this.http.get('https://node-js-servidor-production.up.railway.app/api/obtenerChatRooms/' + id, {responseType: 'text'});
};
```

Figura 3.76: Consumo del método para listar los chat-rooms
Fuente: Elaborado por el investigador

El método enviar mensaje permite enviar el id del usuario emisor, id del usuario receptor y el mensaje para guardarlo.

```
enviarMensaje(emisorId: number, receptorId: number, texto: string): Observable<any> {
  return this.http.post('https://node-js-servidor-production.up.railway.app/api/enviarMensaje', {
    usuarioEmisor:emisorId,
    usuarioReceptor: receptorId,
    mensaje: texto
  }, httpOptions);
};
```

Figura 3.77: Consumo del método para enviar mensaje
Fuente: Elaborado por el investigador

El método editar estado de mensaje permite cambiar el campo leído de un mensaje.

```
aditarEstadoMensaje(emisorId: number, chatRoomId: number): Observable<any> {
  return this.http.put('https://node-js-servidor-production.up.railway.app/api/actualizarMensajes', {
      usuarioEmisor: emisorId,
      id: chatRoomId
  }, httpOptions);
}
```

Figura 3.78: Consumo del método para editar el estado del mensaje
Fuente: Elaborado por el investigador

El método contar mensajes nuevos permite consumir el método para contar los mensajes nuevos que un usuario tiene.

```
contarMensajesNuevos(id:number): Observable<any>{
  return this.http.get('https://node-js-servidor-production.up.railway.app/api/contarNuevosMensajes/' + id, {responseType: 'text'});
}
```

Figura 3.79: Consumo del método para contar mensajes nuevos del usuario
Fuente: Elaborado por el investigador

5. **Métodos de la suscripción.**

   El siguiente método permite consumir el método del api para listas suscripciones.

```
obtenerSuscripciones(): Observable<any> {
 return this.http.get('https://node-js-servidor-production.up.railway.app/obtenerSuscripciones', { responseType: 'text' });
}
```

Figura 3.80: Consumo del método para listar suscripciones
Fuente: Elaborado por el investigador

El siguiente método permite consumir el método del api para guardar la suscripción de un usuario enviando la fecha inicial, fecha final, id del usuario e id de la suscripción.

```
guardarSuscripcionUsuario(usuario:any, suscripcion:any, tipo: any): Observable<any> {
  var CurrentDate = new Date();
  var fechaFin = CurrentDate;
  let mesActual = CurrentDate.getMonth();
  let añoActual= CurrentDate.getFullYear();

  if(tipo === "Mensual") {···
  } else if(tipo === "Semestral"){···
  } else if(tipo === "Anual"){···
  }

  return this.http.post(API_URL+'guardarUsuarioSuscripcion', {
    usuarioId:usuario,
    suscripcionId: suscripcion,
    fechaInicio: new Date(),
    fechaFin: fechaFin
  }, httpOptions);

};
```

Figura 3.81: Consumo del método para guardar suscripción
Fuente: Elaborado por el investigador

6. **Métodos del video.**

   El siguiente método sirve para consumir el método del api para guardar un video, enviando la url del video, url de la miniatura, título, descripción, id del usuario y el id de los equipos para los cuales fue subido el video.

```
guardarVideo(video: any, imagen: any, usuario:any, equipos:any, titulo: string, descripcion: string): Observable<any> {

  return this.http.post(API_URL+'guardar', {
    url:video,
    imagen: imagen,
    titulo: titulo,
    descripcion: descripcion,
    usuarioId:usuario,
    equipos: equipos
  }, httpOptions);
};
```

Figura 3.82: Consumo del método para guardar video
Fuente: Elaborado por el investigador

El siguiente método permite consumir el método del api para reproducir un video enviando la url del video.

```
reproducirVideo(video:any): Observable<any> {
  return this.http.get(API_URL + "reproducir", video);
}
```

Figura 3.83: Consumo del método para reproducir video
Fuente: Elaborado por el investigador

Los métodos obtener videos y listar videos permite consumir los métodos del api para listar los videos que han sido subidos para un equipo específico y listar los video de un usuario, respectivamente.

```
obtenerVideos(equipoId: number): Observable<any> {
    return this.http.get(API_URL+'obtenerVideos/' +equipoId, { responseType: 'text' });

}

listarVideos(usuarioId: number): Observable<any> {
 return this.http.get(API_URL+'listarVideos/' +usuarioId, { responseType: 'text' });

}
```

Figura 3.84: Consumo de los métodos para listar videos
Fuente: Elaborado por el investigador

Los métodos de buscar videos y listar videos por filtro permite consumir los métodos del api para buscar videos según el nombre de usuario, apellido de usuario, título o descripción y listar videos según el título respectivamente.

```
buscarVideos(equipoId: number, texto:string): Observable<any> {
 return this.http.get(API_URL + 'buscarVideos/'+equipoId+ '/' + texto, { responseType: 'text'});
}
listarVideosPorFiltro(equipoId: number, texto:string): Observable<any> {
 return this.http.get(API_URL + 'buscarVideosPorTipo/'+equipoId+ '/' + texto, { responseType: 'text'});
}
```

Figura 3.85: Consumo del método para buscar videos
Fuente: Elaborado por el investigador

El siguiente método permite consumir el método del api para listar los videos no vistos o nuevos.

```
obtenerVideosNoVistos(equipoId: number): Observable<any> {
  return this.http.get(API_URL + 'listarVideosNoVistos/'+equipoId, { responseType: 'text'});
 }
```

Figura 3.86: Consumo del método para listar videos no vistos
Fuente: Elaborado por el investigador

El siguiente método permite el consumo del método de listar videos vistos.

```
obtenerVideosVistos(equipoId: number): Observable<any> {
 return this.http.get(API_URL + 'listarVideosVistos/'+equipoId, { responseType: 'text'});
}
```

Figura 3.87: Consumo del método para listar videos vistos
Fuente: Elaborado por el investigador

El siguiente método consume el método para actualizar el estado del video.

```
aditarEstadoVideo(id: number, equipo: number): Observable<any> {
 return this.http.put(API_URL + 'actualizarEstado', {
    videoId: id,
    equipoId: equipo
 }, httpOptions);
```

Figura 3.88: Consumo del método para actualizar el estado de video
Fuente: Elaborado por el investigador

3.2.4. Fase 4: Pruebas

Tabla 3.63: Prueba de aceptación 1

Prueba de aceptación	**Número: 1**
N° de historia de usuario: 004	
Nombre: Inicio de sesión	
Descripción: ventana que permite a los usuarios acceder al sitio web.	
Condiciones de ejecución: El usuario digitará su correo y clave con la que se creó la cuenta para poder acceder al sitio web.	
Interfaz: La pantalla contiene los campos de email y contraseña, además del boton para realizar la acción y una referencia para la ventana de registro de cuenta .	
Resultado esperado: Una vez que el usuario digite su clave y contraseña, el sistema validará los datos y si son correctos le enviará a la ventana correspondiente según el rol del usuario que inicio sesión.	
Resultado de la prueba: Prueba satisfactoria	

Fuente: Elaborado por el investigador

Tabla 3.64: Prueba de aceptación 2

Prueba de aceptación	**Número: 2**
N° de historia de usuario: 005	
Nombre: Registro de usuarios	
Descripción: ventana que permite a los usuarios crearse una cuenta en el sitio web.	
Condiciones de ejecución: El usuario digitará su información general como nombre, apellido, fecha de nacimiento, género, dirección, teléfono, correo y clave para crear su cuenta y deberá aceptar los términos y condiciones de uso del sistema.	
Interfaz: La pantalla contiene los campos necesarios para que el usuario digite su información básica, un botón para visulizar los términos y condiciones para posteriormente aceptarlos, además tiene un vínculo hacia la ventana de inicio de sesión.	
Resultado esperado: Una vez que el usuario digite la información solicitada para la creación de la cuenta el sistema valida que este completa y guarda la información del usuario.	
Resultado de la prueba: Prueba satisfactoria	

Fuente: Elaborado por el investigador

Tabla 3.65: Prueba de aceptación 3

Prueba de aceptación	**Número: 3**
N° de historia de usuario: 006	
Nombre: Módulo administrador	
Descripción: Este módulo permite al usuario administrador registar equipos y representantes de equipos.	
Condiciones de ejecución: El usuario administrador podrá registrar equipos asignandole su respectiva serie y también registrar representantes de equipos.	
Interfaz: Este módulo contiene dos ventanas; en la ventana para el registro de equipo se puede visualizar un select de las series de equipo y los campos para digitar la información del equipo como el nombre, dirección y teléfono; la ventana para el registro de representante de equipo contiene los campos de nombre, apellido, teléfono, dirección, correo, clave y un select de los equipos existentes en el sistema para asignarle al representante creado.	
Resultado esperado: En la ventana de registro de equipo el usuario administrador digitará la información del equipo y esta se guardará en la base de datos; en la ventana de registro de representante de equipo el administrador digitará la información básica del usuario y la guardará en la base de datos.	
Resultado de la prueba: Prueba satisfactoria	

Fuente: Elaborado por el investigador

Tabla 3.66: Prueba de aceptación 4

Prueba de aceptación	**Número: 4**
N° de historia de usuario: 007	
Nombre: Módulo usuario	
Descripción: Módulo en el que el usuario podrá visualizar información de los equipos existentes, realizar pagos, subir videos y listar sus videos subidos.	
Condiciones de ejecución: El usuario subirá un video y seleccionará el o los equipos para los que quiera enviar su video una vez que realice el pago.	
Interfaz: El módulo de usuario tiene dos ventanas; la primera ventana muestra información sobre los equipos y un formulario para subir un video, la segunda ventana muestra todos los videos que el usuario ha subido	
Resultado esperado: El usuario completará el formulario para subir el video y realizara el pago según el número de equipos seleccionados, una vez realizado el pago el usuario subirá el video y este se guardara en el servidor y en la base de datos.	
Resultado de la prueba: Prueba satisfactoria	

Fuente: Elaborado por el investigador

Tabla 3.67: Prueba de aceptación 5

Prueba de aceptación	**Número: 5**
N° de historia de usuario: 009	
Nombre: Módulo representante de equipo	
Descripción: Módulo que permite al representante de equipo visualizar todos los videos que hayan sido subidos para su equipo, además de buscar y filtrar estos videos.	
Condiciones de ejecución: El representante del equipo deberá pagar una suscripción para poder reproducir el video de su elección.	
Interfaz: El módulo contiene las imágenes de los videos, además de un buscar y filtros.	
Resultado esperado: Una vez que el representante de equipo se haya suscrito a uno de los planes, la información de la suscripción se guardará en la base de datos y el representante podrá reproducir los videos; en la ventana donde se reproduce el video se puede visualizar la información del usuario que subió el video y el administrador del equipo podrá enviarle un mensaje.	
Resultado de la prueba: Prueba satisfactoria	

Fuente: Elaborado por el investigador

Tabla 3.68: Prueba de aceptación 6

Prueba de aceptación	**Número: 6**
N° de historia de usuario: 010	
Nombre: Método de pago	
Descripción: ventana que permite realizar pagos por paypal.	
Condiciones de ejecución: En esta ventana el usuario podrá realizar pagos por medio de paypal.	
Interfaz: La pantalla contiene una tabla con el resumen del item seleccionado, el precio y el total a pagar; además del botón de pago de paypal.	
Resultado esperado: Una vez que el usuario de clic en el botón de pago de paypal se abrirá la ventana de inicio de sesión de paypal para que el usuario acceda a su cuenta y realice el pago correspondiente.	
Resultado de la prueba: Prueba satisfactoria	

Fuente: Elaborado por el investigador

Tabla 3.69: Prueba de aceptación 7

Prueba de aceptación	**Número: 7**
N° de historia de usuario: 013	
Nombre: Cuenta bloqueada y contenido no autorizado	
Descripción: La venta de contenido no autorizado sirve para controlar cuando los usuarios quieran acceder a ciertos módulos del sistema sin iniciar sesión o cumplir con el perfil; mientras que la ventana de cuenta bloqueada aparecerá cuando el usuario este inactivo.	
Condiciones de ejecución: La venta de contenido no autorizado se mostrará solo cuando los usuarios intenten acceder a ciertos modulos sin cumplir con el perfil, mientras que la ventana de cuenta bloqueada se mostrará cuando el usuario que acceda a su cuenta no este activo.	
Interfaz: La pantalla de contenido no autorizado muestra el mensaje de contenido no autorizado y la pantalla de cuenta bloqueada muestra el mensaje de cuenta bloqueada y un botón con la opción de desbloquear cuenta.	
Resultado esperado: el botón de desbloquear cuenta abrirá la ventana de método de pago y una vez completado el pago la cuenta se activará.	
Resultado de la prueba: Prueba satisfactoria	

Fuente: Elaborado por el investigador

Tabla 3.70: Prueba de aceptación 8

Prueba de aceptación	**Número: 8**
N° de historia de usuario: 014	
Nombre: Tareas programadas	
Descripción: Tareas de actualización del estado de suscripción y del estado del usuario.	
Condiciones de ejecución: Las tareas programadas se ejecutarán de forma automática y validará si la fecha final de la suscripción sea igual a la actual para actualizar el estado de la suscripción, una lo hará todos los días por la noche y la otra tarea se ejecutará el 28 de cada mes en la noche y validará la actividad del usuario durante el mes para verificar si no ha subido ningún video para actualizar el estado del usuario.	
Resultado esperado: La tarea programada para actualizar el estado de suscripción cambiará el valor de suscrito a cero del representante de quipo cuya suscripción haya llegado a su fecha final, mientras que la tarea programada para actualizar el estado del usuario cambiará el valor del estado a 0 si no ha subido ningún video durante ese mes .	
Resultado de la prueba: Prueba satisfactoria	

Fuente: Elaborado por el investigador

Tabla 3.71: Prueba de aceptación 9

Prueba de aceptación	**Número: 9**
N° de historia de usuario: 016	
Nombre: Control de errores	
Descripción: Validar que no existan errores y controlarlos.	
Condiciones de ejecución: controles para campos vacíos, tipos de datos y tiempo del video.	
Resultado esperado: En los formularios no se permitirán campos vacíos y se validarán los tipos de datos que se ingresaran en cada campo, el control del tiempo de video no permitirá subir videos que no tengan 3 minutos de duración.	
Resultado de la prueba: Prueba satisfactoria	

Fuente: Elaborado por el investigador

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

- Se desarrolló un sitio web utilizando tecnologías open source mismas que se detallan en el presente proyecto, dicho sitio permite exhibir las habilidades y talentos de los postulantes por medio de videos que pueden ser visualizados por los directivos de los equipos y a través de mensajería permite el contacto en línea entre postulantes y directivos con el fin de gestionar su selección y reclutamiento.

- Para el análisis de los requerimientos del sitio web para la gestión de la selección y reclutamiento de futbolistas en la ciudad de Ambato se utilizó una entrevista realizada a los directivos de las principales equipos de la cuidad y una ficha para la observación de campo con lo que se pudo obtener como resultados que el sitio debía manejar perfiles de usuarios, soportar subida de videos, pagos, guardar información, reproducción de video, visualización de imágenes y envío de mensajes.

- Después de investigar sobre los frameworks de desarrollo más populares y seleccionar tres de ellos para hacer una comparativa mediante un cuadro en base a sus características, se seleccionó el framework de Angular js para el desarrollo del front-end, ya que al tener un patrón de diseño ya establecido, una gran cantidad de componentes propios y amplia documentación facilitó el diseño y desarrollo del front.

- Una vez culminado el desarrollo y las pruebas de funcionamiento del sitio web, se subió cada parte del proyecto en un host diferente: para el servidor se utilizó el host Railway, mientras que el cliente fue subido a Hostinger. Esto permitirá que el sitio web para la gestión del reclutamiento y selección de futbolistas funcione de forma óptima, ya que separar el cliente del servidor facilita la escalabilidad del sitio web y, a su vez, mejora su seguridad.

4.2. Recomendaciones

- Se recomienda adquirir un plan de hosting pagado para subir el servidor del proyecto realizado debido a que el uso del plan gratuito de railway tiene limitaciones como es el tiempo de ejecución mensual, capacidad de memoria RAM y disco.
- Para el correcto funcionamiento del sitio web se recomienda llevar un control de las fechas de renovación de los planes adquiridos del host y dominio con la finalidad de mantenerlos siempre activos.
- Se recomienda que al usar librerías en angular siempre se utilice la versión mas actualizada para evitar conflictos con las dependencias del proyecto.
- Se sugiere realizar respaldos periódicos de la información de la base de datos, para evitar su pérdida o corrupción.

Bibliografia

[1] C. A. P. Menéndez, "Diseño e implementación de plataforma web para control y seguimiento de actividades del personal del sistema canal radio y televisión de la universidad católica santiago de guayaquil," 2019.

[2] A. O. V. AgustóÃ§in Marcelo De Luca, "Machi: Aplicación móvil para el acercamiento de la tecnología al deporte," 2021.

[3] W. H. B. N. Josué Danilo Delgado Navas, "Propuesta de criterios de selección de talentos en la escalada deportiva," *Revista Arbitrada Interdisciplinaria Koinonía*, vol. 6, no. 2, 2021.

[4] J. S. L. Arenas, "Toma de decisiones en la selección de futbolistas, por parte de los entrenadores y directivos de la agencia prado managers s.a.s, a partir de un modelo de gestión del conocimiento," 2021.

[5] F. G. V. Gallardo, "Desarrollo de aplicación movil para gestionar el proceso de reclutamiento y selección de futbolistas en clubes," 2021.

[6] R. S. Pressman, *Ingeniería de software– Un enfoque práctico*. The McGraw-Hill Companies.

[7] D. de Sistemas, "Metodología de desarrollo de software," 2017.

[8] "React - conceptos básicos," 2019.

[9] R. d. A. Garcia, "Estudio de la popularidad del framework vuejs,"

[10] Sanchez, "Capítulo 3 : Spring, un framework de aplicación,"

[11] "Spring framework," 2018.

[12] M. Cíceri, "Introducción a laravel: Aplicaciones robustas y a gran escala," 2018.

[13] A. Mardan, "Pro-express.js," 2016.

[14] "Express.js," 2022.

[15] "Expressjs," 2017.

[16] R. V. Lerma-Blasco, *Aplicaciones web*. McGraw-Hill España, 2013.

[17] J. Z. Jiménez, "Aplicaciones web," 2013.

[18] P. Parlebas, "Juegos, deporte y sociedades. léxico de praxeología motriz," 2001.

[19] J. R. López, *Deporte y Ciencia - Teoría de la actividad física*. INDE, 2003.

[20] A. A. Molina, M. G. Román, C. I. B. Díaz, B. E. O. Acosta, F. D. Madrazo, C. H. B. Vidaurri, B. E. O. Acosta, M. A. R. Figueroa, and M. C. Ivo], *Selección y desarrollo de talentos deportivos. Una propuesta para el ámbito escolar*. Instituto Tecnoógico de Sonora, 2010.

[21] E. A. P. Santiago, M. Q. López, H. R. O. Aguirre, and V. M. L. Moreno, "Clasificación de jugadores de futbol soccer basada en sus habilidades deportivas, físicas y mentales," 2018.

[22] G.Idoate, "Jugador de fútbol,"