Revista Peruana de Computación y Sistemas 2023 5(2):3-15
https://doi.org/10.15381/rpcs.v5i2.25244

ISSN Online: 2617-2003
Facultad de Ingeniería de Sistemas e Informática
Universidad Nacional Mayor de San Marcos

Artículo de Contribución

Aplicación de repositorio para el club de robótica de la Universidad Técnica de Cotopaxi

Repository application for the robotics club of the Technical University of Cotopaxi

José Cadena [1,a], Juan Carlos Chancúsig [1,b], Jorge Ante [1,c], Bryan Cornejo [1,d], Estefanía Guanoluisa [1,e]

[1] Universidad Técnica de Cotopaxi, Facultad de Ciencias de la Ingeniería y Aplicadas. Latacunga, Ecuador

[a] Autor de correspondencia: jose.cadena@utc.edu.ec, ORCID: https://orcid.org/0000-0002-1775-252X

[b] E-mail: juan.chancusig@utc.edu.ec, ORCID: https://orcid.org/0000-0003-0346-1729

[c] E-mail: jorge.ante9817@utc.edu.ec, ORCID: https://orcid.org/0009-0000-5308-7523

[d] E-mail: bryan.cornejo0498@utc.edu.ec, ORCID: https://orcid.org/0009-0005-0476-0941

[e] E-mail: estefania.guanoluisa2880@utc.edu.ec, ORCID: https://orcid.org/0009-0004-2573-1333

Resumen

Este trabajo resume el diseño de un repositorio de información desarrollada para el club de robótica (BOTS UTC) de la Universidad Técnica de Cotopaxi. Se presentó la fase de desarrollo del aplicativo. Los métodos utilizados y la importancia de las herramientas empleadas para el desarrollo del programa, creación de modelos, diagramas y BDD (base de datos). Las herramientas usadas para el desarrollo del aplicativo se tomaron en cuenta mediante los resultados obtenidos de un estudio previo realizado donde se seleccionaron diversas herramientas metodologías, arquitecturas, SGBD, y lenguajes de programación que usaban actualmente las empresas desarrolladoras de software. Se desarrolló una descripción de la fase de implementación, mantenimiento y pruebas correspondientes al modelo de cascada, en los cuales nos basamos para la elaboración del proyecto. Teniendo en cuenta el objetivo propuesto, este aplicativo fue creado para solucionar problemas en el ámbito educativo y social, beneficiando a los usuarios con esta propuesta.

Palabras clave: software, BDD, repositorio.

Abstract

This work summarizes the design of an information repository developed for the robotics club (BOTS UTC) of the Technical University of Cotopaxi. The development phase of the application was presented. The development phase of the application was presented. The methods used and the importance of the tools used for the development of the program, creation of models, diagrams and BDD (database). The tools used for the development of the application were taken into account through the results obtained from a previous study carried out where various tools, methodologies, architectures, DBMS, and programming languages currently used by software development companies were selected. A description of the implementation, maintenance and testing phase corresponding to the waterfall model was developed, on which we based ourselves for the development of the project. Taking into account the proposed objective, this application was created to solve problems in the educational and social field, benefiting users with this proposal.

Keywords: software, BDD, repository.

Recibido: 22/05/2023 - Aceptado: 25/10/2023 - Publicado: 20/12/2023

Citar como:
Cadena, J., Chancúsig, J., Ante, J., Cornejo, B. & Guanoluisa, E. (2023). Aplicación de repositorio para el club de robótica de la Universidad Técnica de Cotopaxi. Revista Peruana de Computación y Sistemas, 5(2):3-15. https://doi.org/10.15381/rpcs.v5i2.25244

1. Introducción

Actualmente, con el desarrollo de las TICs se han propuesto nuevas soluciones óptimas para las empresas mediante el uso de software. Por ello, se realizará un análisis sobre el desarrollo y creación del aplicativo BOTS UTC, una actividad a cargo del proyecto formativo de la Universidad Técnica de Cotopaxi, una propuesta desarrollada por integrantes en el área de prácticas laborales.

Actualmente en el club de Robótica de la Universidad Técnica de Cotopaxi, el proceso de gestión documental se maneja manualmente, por lo que hubo la necesidad de informatizar dicho proceso. Este artículo surge como producto del proyecto "Repositorio de Información para el club de Robótica". En el presente documento se describirán, las herramientas utilizadas para la elaboración del MER (Modelo entidad-relación) en MySQL, diagrama de clases UML, modelo lógico y físico MySQL. Además de los métodos utilizados para la programación del aplicativo. Por otro lado, se realiza un análisis del uso del lenguaje Javascript [20], descripción breve de Netbeans [21], la herramienta Xampp [22] y MySQL [23].

Nuestro propósito es indicar los resultados obtenidos al haber finalizado con el desarrollo del proyecto, desde la parte del análisis, diseño, desarrollo, implementación y pruebas. Los resultados que se presentan a continuación reflejan parte de la funcionalidad del aplicativo y sus componentes.

En cuanto a la estructura del documento, el mismo contempla una breve introducción sobre la temática, continuando por una revisión profunda de la literatura, para luego avanzar con la metodología propuesta ya en la elaboración del aplicativo. A continuación, se presenta los resultados obtenidos para entrar luego a las conclusiones respectivas y finalmente terminar con las referencias bibliográficas.

2. Revisión de Literatura

2.1. Arquitecturas de Software

Tal cual la arquitectura es una técnica que se utiliza para poder, en su mayoría, diseñar y hacer proyecciones a futuro de los edificios; la arquitectura de software es una técnica que nos sirve para modelar, diseñar y programar un software específico dependiendo de los requerimientos y la protección que se tenga del mismo. Según Ford [11] "Los desarrolladores se han esforzado durante mucho tiempo por acuñar una definición sucinta y concisa de la arquitectura del software, ya que su alcance es amplio y está en constante cambio." La arquitectura de software como tal aún no tiene una definición concreta ya que, al tratarse de algo relacionado con el mundo de la tecnología, este se encuentra en constante evolución debido a las nuevas tecnologías y herramientas que surgen con el paso del tiempo.

Dicho esto, "Ralph Johnson definió famosamente la arquitectura de software como "lo importante (sea lo que sea)". El trabajo del arquitecto es comprender y equilibrar todas esas cosas importantes (sean las que sean). Ford [11]"

Una parte importante dentro de la arquitectura de software es los requerimientos del negocio y la persona a cargo (el arquitecto) debe encargarse de entender dichos requisitos, junto a otros factores, para poder dar solución a una propuesta.

Los factores arquitectónicos pueden ser Ford [11]:

- Auditability (Auditabilidad) = capacidad de una auditoría para otorgar datos resultados precisos.
- Performance (Rendimiento).
- Security (Seguridad).
- Requirements (Requisitos).
- Data (Datos-Información).
- Legality (Legalidad).
- Scalability (Escalabilidad) = capacidad de adaptación y respuesta de un sistema con respecto al rendimiento.

2.2. Tipos de arquitectura de Software

Al existir diferentes factores a tomar en cuenta y según sea la propuesta dada, un arquitecto puede escoger entre diferentes tipos de arquitecturas de software ya establecidas y probadas por diferentes desarrolladores, las cuales han demostrado un grado alto de eficacia al momento de desarrollar y ejecutar un software. Entre las que existen podemos nombrar:

Arquitectura Ágil

Según Navarro [12], "En la Arquitectura Ágil se enfatiza fuertemente en el concepto de los Requisitos Significantes para la Arquitectura, los cuales son aquellos requisitos que tienen un impacto medible en una arquitectura de software".

El término Arquitectura Ágil nace de la unión de una Metodología Ágil (como puede ser SCRUM) y de la arquitectura de software como tal; implementado así a la arquitectura de software dentro del desarrollo de la metodología. Permitiéndonos de esta manera diseñar una arquitectura que sea: fácil de probar, ambientar, instalar, entregar, etc. Logrando conseguir un diseño técnico que sea evolutivo.

Basada en componentes

Según Hernández [13], "Desde un punto de vista orientado a objetos, un componente es un conjunto de clases que colaboran, por lo que en cada script de la arquitectura de software se definen las clases requeridas para definirlo como componente"

Explicado de otra manera, la Arquitectura Basada en componentes busca colaborar con una Arquitectura en capas otorgando componentes funcionales y lógicos que poseen una interfaz de comunicación bien definida.

Arquitectura dirigida por eventos (EDA)

Según Ford [11], "Las arquitecturas basadas en eventos (EDA) suelen integrar varios sistemas dispares utilizando colas de mensajes. Hay dos implementaciones comunes de este tipo de arquitectura: los patrones broker y mediador."

Como dice en el nombre, este tipo de arquitectura se basa en eventos que toman lugar durante el funcionamiento del software, y que generan un cambio en el mismo. Para ello el software debe capturar y procesar dichos eventos mediante las implementaciones:

- Broker: El cual es un programa intermediario que se encarga de traducir los mensajes de un sistema desde un lenguaje a otro, el cual posee Ford [11]: colas de mensajes, evento iniciador, eventos intra-procesos y procesadores de eventos.
- Mediador: Es el que se encarga de la coordinación y enviar mensajes a la cola de mensajes para activar los procesadores de eventos.

Arquitectura orientada a servicios (SOA)

Según Ford [11] "Las arquitecturas ESB suelen utilizar los mismos bloques de construcción que las EDA, la organización de los servicios difiere y se basa en una taxonomía de servicios estrictamente definida. El estilo ESB difiere de una organización a otra, pero todas se basan en la segregación de los servicios en función de la reutilización, los conceptos compartidos y el alcance."

Las ESB es una arquitectura orientada a servicios que utiliza una coordinación a través de un bus de servicio (Enterprice Service Bus, ESB), el bus de servicios actúa como mediador para las interacciones de eventos complejos y se encarga de otras tareas típicas de la arquitectura de la EDA ya que se suelen utilizar los mismos bloques de construcción. En este contexto podemos decir que un servicio es una unidad autónoma de una o más funciones del software diseñada para realizar una tarea específica

Arquitectura sin servidor

Según Ford [11], "El otro tipo de arquitectura sin servidor es la FaaS (Function as a Service), que evita por completo la infraestructura (al menos desde el punto de vista del desarrollador), aprovisionando la infraestructura por solicitud, gestionando automáticamente el escalado, el aprovisionamiento y una serie de otras tareas de gestión."

Una arquitectura sin servidor se refiere a cuando un desarrollador busca contratar servicios de terceros para poder desarrollar su proyecto, ahorrando el costo de mantener un servidor propio y programar servicios que se necesitan para que este funcione correctamente. Las funciones de FaaS se activan mediante tipos de eventos definidos por el proveedor de servicios.

Arquitectura Monolítica

Según Ford [11] "Las arquitecturas monolíticas suelen contener una gran cantidad de código altamente acoplado" Esto quiere decir todas las funciones y servicios se agrupan dentro de un componente único y centralizado. Con el paso del tiempo y la aparición de proyectos más grandes y con procesos más delicados, este tipo de arquitectura ha ido quedando en el olvido.

Big Ball of Mud

"Consideremos el caso degenerado de un sistema caótico sin arquitectura discernible, conocido coloquialmente como el antipatrón de la Gran Bola de Barro. Aunque existen elementos arquitectónicos típicos, como marcos y bibliotecas, los desarrolladores no han construido la estructura a propósito… Los desarrolladores crearon clases altamente acopladas con modularidad escasa. Ford [11]"

Este tipo de arquitectura se genera involuntariamente por parte del desarrollador ya que este último, ya sea por presión o falta de tiempo, crea un sistema con una estructura caótica y con un crecimiento descontrolado con todas sus clases y módulos mezclados entre sí. Por lo tanto, desde el punto de vista de la evolucionabilidad, esta arquitectura tiene una puntuación extremadamente baja. Los desarrolladores que necesitan cambiar el acceso a los datos en toda la aplicación deben buscar todos los lugares en los que existe y cambiarlos, arriesgándose a perder algunos lugares.

Onion Architecture

Según [14] "Jeffrey Palermo propuso la denominada Onion Architecture o arquitectura en forma de cebolla, la cual reestructura el modelo de arquitectura encírculos, siendo estos círculos las áreas del modelo de arquitectura"

La propuesta principal de esta arquitectura es el buen acoplamiento. La regla fundamental es que todo el código puede depender de las capas más centrales, pero el código no puede depender de las capas más alejadas del núcleo. En otras palabras, todo el acoplamiento es hacia el centro. Esta arquitectura se inclina descaradamente hacia la programación orientada a objetos, y antepone los objetos a todos los demás.

Clean Architecture

Según [14] "Clean Architecture es una idea de modelo de arquitectura que propone el desarrollador Robert C. Martin, el mismo que propuso aplicar los conceptos SOLID al desarrollo de software."

La idea principal de este tipo de arquitectura es que cada nivel pueda realizar sus propias funciones y que se comunique únicamente con los niveles contiguos.

Arquitectura basada en capas

Este tipo de arquitectura se basa en la distribución de responsabilidades de una forma jerárquica de modo que es más ordenado y optimo la responsabilidad que cada capa adquiere, es por este motivo que este tipo de arquitectura es de las más utilizadas, generalmente esta arquitectura es implementada en 4 capas que se las podría definir como Capa de presentación, Capa de negocios, Capa de persistencias y Capa de Base de Datos [15]. Sin embargo, no todas las aplicaciones solo se mantendrán en estas cuatro capas, dependiendo de las necesidades de cada aplicación se podrán incrementar estas capas.

La manera en cómo se estructura la arquitectura de capas es de modo horizontal de modo que cada capa solo puede comunicarse con la capa que se encuentra inmediatamente por debajo de la misma es decir que si la capa de reglas de negocio trata de comunicarse u obtener información de la capa de base de datos esta tendrá que primero interactuar con la capa de persistencia, otras características que pueden identificar a la arquitectura basada en capas son las siguientes [16]:

- Descripción: Denota la composición de servicios de tal forma que gran cantidad de las interacciones ocurre únicamente entre capas continuas.
- Residencia: Las capas de una aplicación pueden alojarse en la misma maquina física o misma capa, de igual forma puede estar distribuido sobre diferentes computadores.
- Comunicación: Los elementos que conforman cada capa tienen la capacidad de comunicarse con otros elementos ubicados en diferentes capas mediante interfaces muy bien establecidas.
- "Pirámide invertida de re-uso": Se lo conoce así debido a que por su estructura y forma de uso a cada capa se le agrega cierta responsabilidad de igual manera un proceso de abstracción con la capa directamente ubicada encima de ella.

Microfrontends

De acuerdo a Camarena [18] Los microfrontends son una técnica de arquitectura de software que consiste en dividir una aplicación frontend monolítica en componentes más pequeños e independientes.

Domain Driven Design (DDD)

Según [19], El Domain Driven Design (DDD) es un enfoque de diseño de software que se centra en el dominio del problema y en la colaboración entre expertos en el dominio y desarrolladores de software.

Monolito modular

Un monolito modular es una arquitectura de software en la que una aplicación se desarrolla como una sola unidad monolítica, pero se organiza en módulos independientes y acoplados [18], [19].[3]

Cliente-Servidor

[15] Nos dice que, este tipo de arquitectura es conocida por ser muy utilizada a nivel comercial, este se encuentra estructurado principalmente por dos componentes muy específicos el proveedor y el consumidor dentro de estos dos componentes debemos entender que el proveedor es un servidor que brinda una gran cantidad de servicios o recursos los cuales son necesitados e implementados por el cliente, por otra parte podemos poner en evidencia ciertos componentes extra que proporcionan a esta arquitectura una estructura más confiable entre estas podemos encontrar [17]:

- Red: Este componente se conforma por los servidores, bases de datos y un conjunto de clientes que se encuentran agrupados de una manera física o no física dentro de los cuales existen ciertos protocolos los cuales facilitan la transmisión de información ya establecida previamente
- Cliente: A este componente lo podemos tomar como el elemento que requiere los servicios, este puede ser un computador o en si la aplicación informática dentro de los cuales se solicitara diferente información para funcionar la cual se obtendrá a través de la red.
- Servidor: Este componente va a ser quien ofrezca los diferentes servicios al cliente y de igual forma que este puede ser un computador o la aplicación informática, sin embargo, este será quien almacena la información y la transmite al cliente por medio de la red.
- Protocolo: A este componente lo podemos tomar como el reglamento dentro de esta arquitectura ya que dentro de este se encuentra el conjunto de normas o procedimientos establecidos de manera clara y concreta acerca de cómo la información transitara a través de la red hacia sus diferentes destinos.
- Servicios: Este componente se encarga de responder los requerimientos que el cliente necesite ya que posee un gran conjunto de información el cual ayudara a la solución de sus necesidades.
- Base de Datos: Este componente se encuentra conformado por diferentes conjuntos de información correctamente ordenada la cual se va a encontrar dentro de la red y que se encontrará a total disponibilidad del servidor o directamente podrá interactuar con los clientes.

3. Metodología

Para la creación de la base de datos del proyecto inicialmente se seleccionó varias herramientas entre las cuales sobresalieron 2 esta son Xampp y Wamp, aunque ya que ambas cuentan con atributos similares a continuación se detalla la comparación que se realizó entre las dos (Ver Tabla 1), y de esta forma determinar la que mejor se acople a nuestras necesidades.

Tabla 1

Comparación entre Xampp y Wamp

XAMPP	WAMP
Su proceso de instalación es más sencillo que el de Wamp.	El proceso de modificar las configuraciones es más fácil e intuitivo de realizar en Wamp.
Cuenta con una gran cantidad de módulos entre los que se destaca Joomla, OpenSSL entre otros.	La creación y codificación de base de datos en Windows es más fácil realizar a comparación de Xampp.
Cuenta tanto con su versión estándar y por otro lado está disponible en su versión completa.	Está disponible tanto para sistemas de 32 y 64 Bits.
Ya que es multiplataforma se puede instalar en cualquier sistema operativo ya sea Windows, Linux o Mac.	
Con simples pasos se puede tanto iniciar como detener tanto el servidor como la base de datos.	

Como podemos evidenciar después de haber realizado la comparación entre Xampp y Wamp se ha podido hacer la elección de la herramienta Xampp, el mismo que a nuestro criterio es la que más se acopla a nuestras necesidades y nos permite desarrollar el proyecto de una manera más eficiente.

Detalles de Xampp

Xampp es una herramienta la cual está apenas iniciándose en el mundo de la programación y el desarrollo web ya que presenta una interfaz fácil de manejar hoy en la actualidad Xampp es uno de los servidores más utilizados mundialmente para trabajar en PHP y Perl, esto se debe a que es muy práctico y de fácil instalación, Además gracias a la compatibilidad con la que cuenta se lo puede instalar con todos los sistemas operativos existentes [1].

Xampp no es simplemente un programa, sino que al contrario es un paquete el cual está compuesto por varios softwares dedicados al manejo y gestión de base de datos. Fue desarrollado por Apache Friends y su nombre esté compuesto por las iniciales de cada uno de los softwares y herramientas que componen Xampp, estos son:

X: La primera inicial es la representación de todos los sistemas operativos para los cuales está disponible entre estos están Windows, Ubuntu, Linux, Mac entre otros.

A: Esta pertenece al servidor de apache ya que proporciona diferentes herramientas las cuales ayudan a trabajar con este software.

M: La letra M por otro lado indica que Xampp no incorpora en sus funciones el sistema de gestión de base de datos MySQL.

P: Indica que Xampp utiliza PHP como su lenguaje de programación ya que este es muy conocido y además es compatible y soportado por una gran cantidad de sistemas de base de datos.

P: Finalmente la P hace referencia a otro lenguaje de programación con el que trabaja Xampp el cual es Perl a diferencia de PHP este se centra en la administración del sistema y de la programación de red, como se indica en la Fig. 1

Base de datos MySQL

Dentro de Xampp hemos optado por utilizar la herramienta de MySQL para realizar la base de datos en la que se basará el proyecto para poder realizar las inserciones y el registro de los datos ingresados desde el aplicativo.

Fig 1

Interfaz Xampp.

MySQL a nuestro parecer es una la base de datos más sencilla y fácil de utilizar ya que es una de las bases de datos más utilizada a nivel mundial debido a que cuenta con una gran cantidad de beneficios por lo que incluso es utilizada por grandes compañías como lo son: Facebook, Twitter e incluso YouTube.

MySQL cuenta con varias características importantes entre las que hemos destacado las siguientes:

- Permite realizar consultas en la base de datos de una manera rápida.
- Es compatible con casi todos los sistemas operativos existentes como Windows, Mac, Linux entre otros.
- Nos permite realizar cambios en la base de datos mediante comandos DCL, DML y DDL.
- Es considerado uno de los softwares más rápidos y eficientes actualmente.
- Gracias a la gran cantidad de herramientas con las que cuenta facilita en gran medida su uso [6].

Finalmente, la razón por la cual hemos decidido trabajar con MySQL es que nos proporciona varios

beneficios entre los que se resaltan los siguientes.

- Es escalable y de alto rendimiento
- Es open Source
- Tiene alta disponibilidad.
- Es fácil de utilizar y configurar [2].

Presentamos una interfaz de datos MySql, Fig. 2.

Antes de iniciar con el desarrollo del aplicativo, se propuso realizar un estudio de factibilidad y viabilidad previo a la etapa de análisis, mediante la aplicación de encuestas realizadas al club de robótica. Los resultados obtenidos reflejaron la necesidad de un repositorio de información que les permitiera almacenar datos importantes del club, información de los integrantes, archivos e inventario de los dispositivos adquiridos.

Para ello, las metodologías usadas corresponden a los resultados obtenidos en base a un estudio previo de las arquitecturas, metodologías, SGBD, lenguajes de programación que se utilizan mayormente en empresas desarrolladoras de software, se realizó resumen comparativo entre el MVC y SOA, el mismo que se presenta en la Tabla 2.

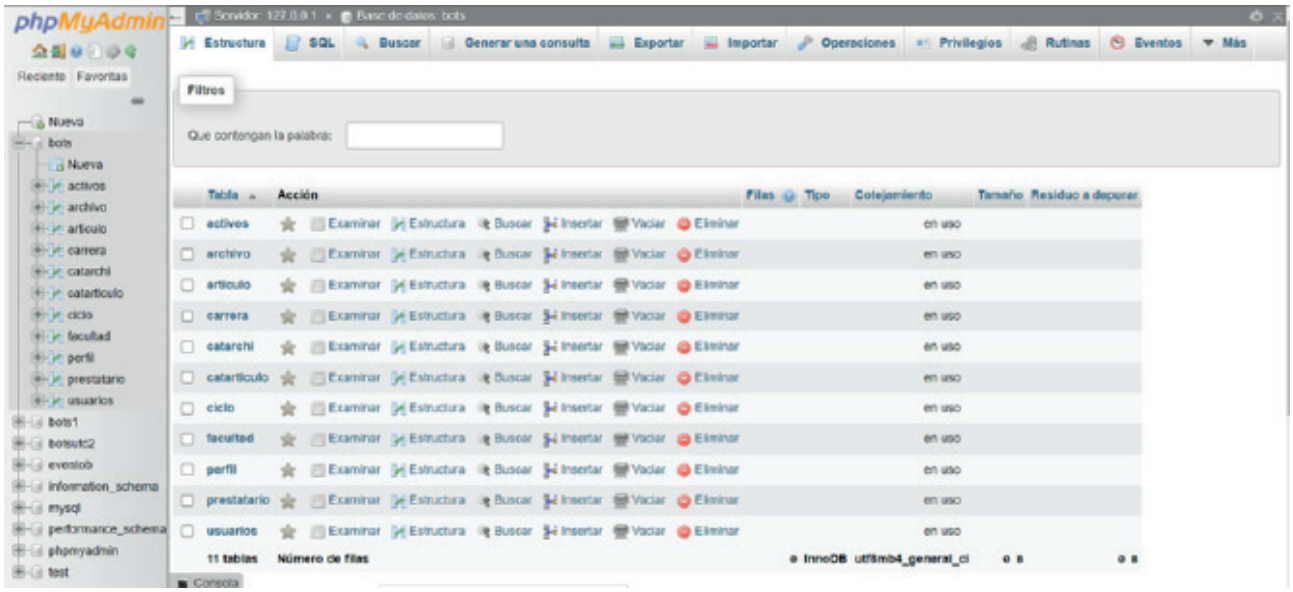
Tabla 2

Cuadro comparativo de arquitecturas

	Modelo Vista Controlador (MVC)	Arquitectura Orientada a servicios (SOA)
Componentes	Lógica de negocio y Base de Datos Interfaz Gráfica de Usuario (GUI) Controlador de Acciones	Definición Desarrollo Evolución Distribución
Características	Separación de las tareas de aplicación Capacidad de prueba Desarrollo controlado por pruebas Marco extensible y conectable	Reutilización, Interoperabilidad, Escalabilidad, Flexibilidad, Eficiencia en coste
Funciones	Solicitar datos Comunicar datos Enviar datos Mostrar datos visualmente	Proveedor de servicios Agente de servicios Usuario del servicio

Fig 2

Interfaz base de datos Mysql.



Arquitectura utilizada

Para el desarrollo de este proyecto se utilizó la arquitectura Modelo Vista Controlador (MVC) la cual, es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, y sobre multitud de lenguajes y plataformas de desarrollo [3] (ver Figura 3).

Modelo: Maneja datos y lógica de negocios.

Vista: Se encarga del diseño y presentación.

Controlador: Enruta comandos a los modelos y vistas.

El modelo es el responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".
- Lleva un registro de las vistas y controladores del sistema.
- Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero por lotes que actualiza los datos, un temporizador que desencadena una inserción, etc.).

El controlador es responsable de:

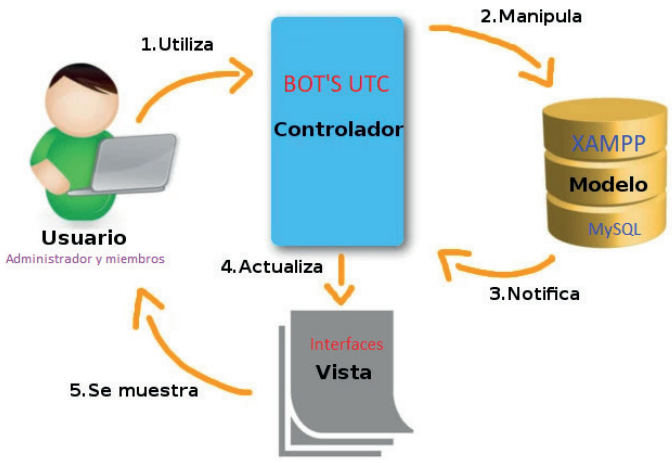
- Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar()". Una petición al modelo puede ser "Obtener_tiempo_de_entrega (nueva_orden_de_venta)".
- Las vistas son responsables de:
- Recibe datos del modelo y los muestra al usuario.
- Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
- Pueden dar el servicio de "Actualización()", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

Beneficios de utilizar MVC:

- Las principales ventajas de hacer uso del patrón de arquitectura MVC son:
- La separación del Modelo de la Vista, es decir, separar los datos de la representación visual de los mismos.
- Es mucho más sencillo agregar múltiples representaciones de los mismos datos o información.
- Facilita agregar nuevos tipos de datos según sea requerido por la aplicación ya que son independientes del funcionamiento de las otras capas.

Fig 3

MVC



- Crea independencia de funcionamiento.
- Facilita el mantenimiento en casos de errores.
- Las desventajas de seguir el planteamiento de MCV son:
- La separación de conceptos en capas agrega complejidad al sistema.
- La cantidad de archivos a mantener y desarrollar se incrementa considerablemente.
- La curva de aprendizaje del patrón de diseño es más alta usando otros modelos más sencillos.

Para ello, hemos elaborado el modelado del club de robótica como se muestra en la Figura 4.

Para lograr almacenar los datos se creó una base de datos en el SGBD MySQL, fue seleccionado por el rendimiento, flexibilidad y velocidad. Que permitió conectarse con el servidor Apache y MySQL mediante phpMyAdmin y administrar de forma sencilla la base de datos.

Oracle SQL Data Modeler[4], es una de las herramientas proporcionadas por Oracle la cual posee una interfaz gráfica que permite diseñar modelos relacionales, lógicos, físicos, etc. Este gestor de modelos ofrece la capacidad de realizar una ingeniería directa y/o inversa lo cual genera una productividad muy alta dentro de los usuarios que requieren realizar un boceto de sus bases de datos y sin duda será utilizada dentro de varios proyectos en un futuro ya que como nos menciona [5] los sistemas basados en BI (*Business Intelligence*) tienen la necesidad de recoger y presentar un sin número de información que recolecta un negocio.

Por lo tanto, su objetivo es, ayudar en la toma de decisiones para que estos sistemas funcionen correctamente, por ello, se requiere de modelos de datos relacionales bien especificados por lo cual el uso de la herramienta de Oracle Data Modeler ayudará a los diseñadores de base de datos a tener una vista detallada de cómo se relacionan sus esquemas para generar información útil para una organización que se maneje bajo BI.

Un modelo lógico contiene representaciones de entidades y atributos, relaciones, identificadores exclusivos, subtipos y supertipos y restricciones entre relaciones [6].

Un modelo de datos físicos es un modelo específico de base de datos que representa objetos de datos relacionales (como tablas, columnas, claves principales y claves externas) y sus relaciones. El modelo de datos físicos se puede utilizar para crear sentencias DDL que se distribuyen a los servidores de bases de datos [7].

Mediante el uso de esta herramienta se realizó el modelado lógico y físico, como se muestra en la figura 5 y 6.

El desarrollo del aplicativo es orientado a datos (data-driven) por lo que se inicia con el diseño de base de datos. Con la creación de la base de datos, se inició la elaboración del diagrama de clases, el cual se muestra en la Figura 7.

Se realizaron algunos bocetos en Balsamiq para determinar las interfaces requeridas por el usuario. De

Fig 4.
Modelo lógico en MySQL.

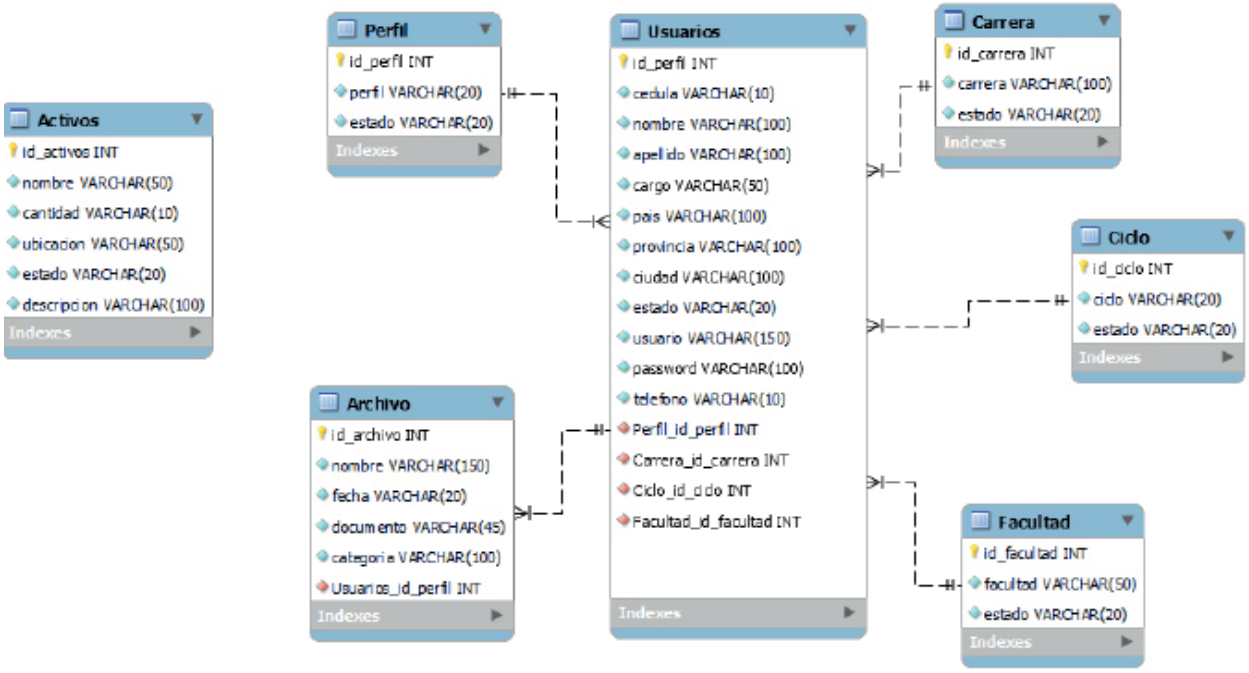


Fig 5.
Modelo lógico en Oracle

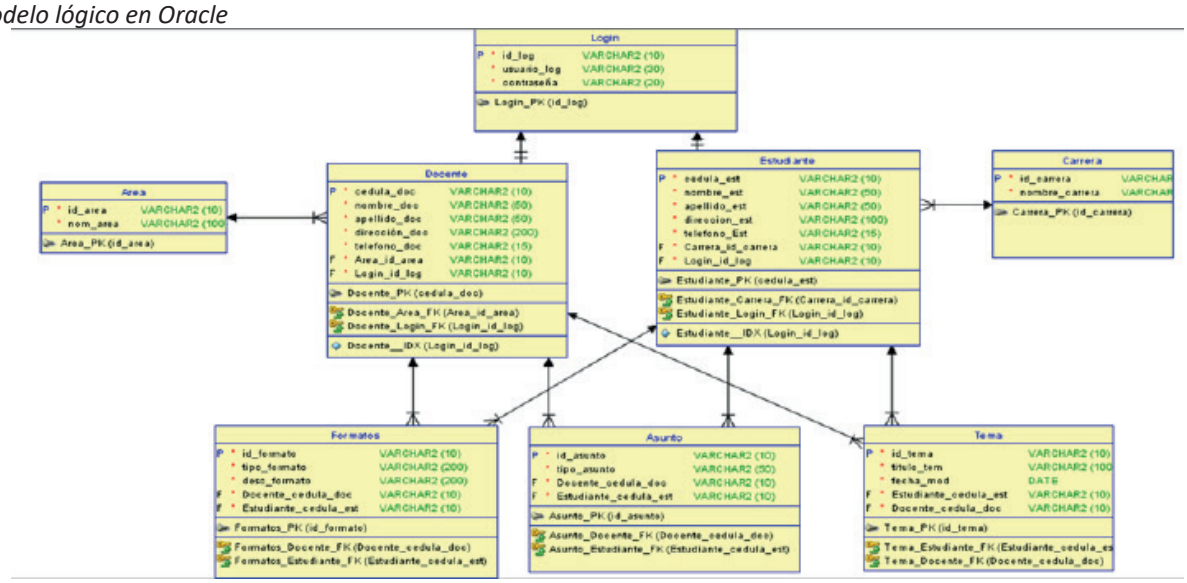
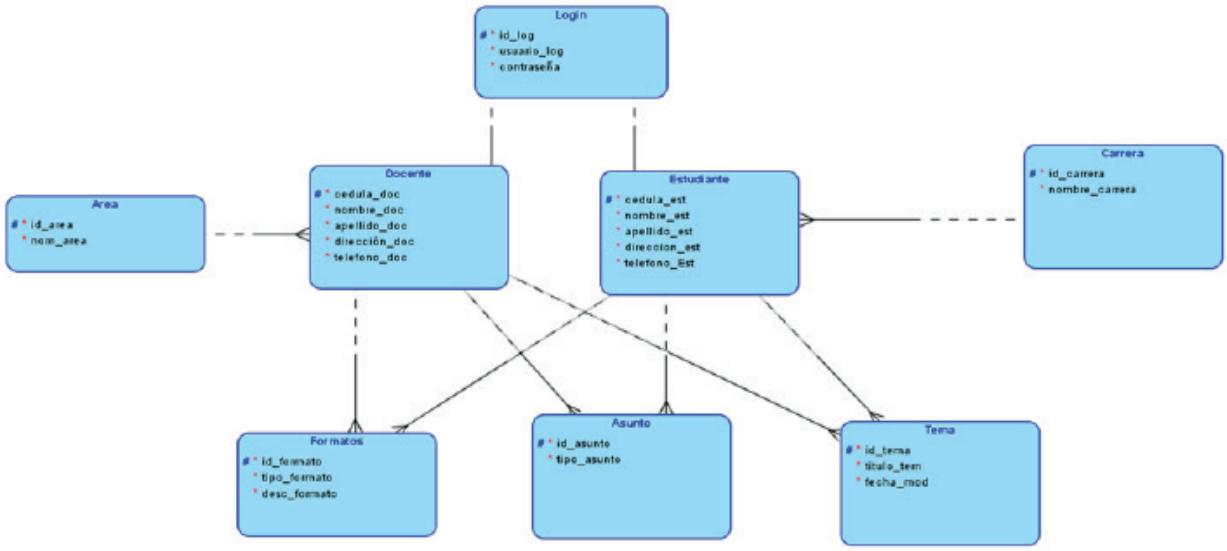


Fig 6.
Modelo físico en Oracle



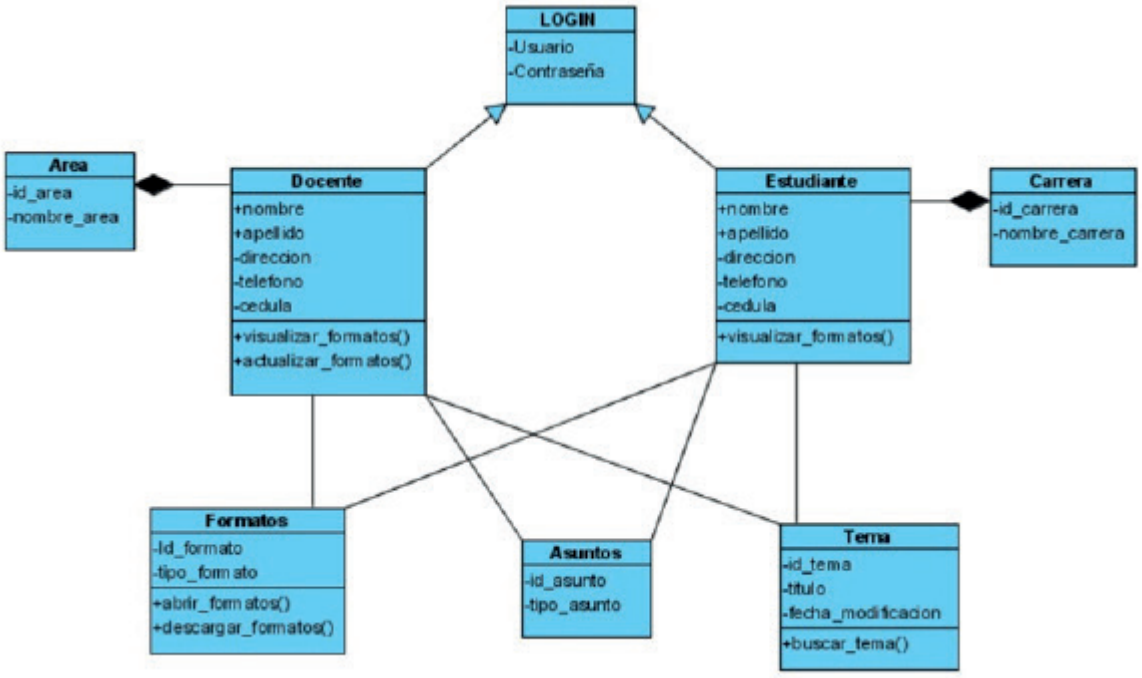
acuerdo al diagrama de clases presentado en la Figura 7, se inició con la programación de cada una de las secciones. Es necesario indicar que la multiplicidad en un diagrama de clases indica la cantidad de instancias que pueden existir en una relación entre dos clases. Se representa mediante números o rangos que indican la cantidad mínima y máxima de instancias que pueden existir.

Se ha pospuesto utilizar Java, porque es un lenguaje de programación libre, el cual ofrece al programador una arquitectura muy organizada, para que el usuario pueda visualizar el funcionamiento de cada parte del código [8].

Mediante el uso del lenguaje de programación Java se desarrolló el código y diseño los formularios del entorno del aplicativo BOT'S UTC de cada una de las secciones identificadas: inicio, miembros, activos y archivos.

Netbeans es una herramienta que trabaja junto al lenguaje de programación Java[9], este software tiene

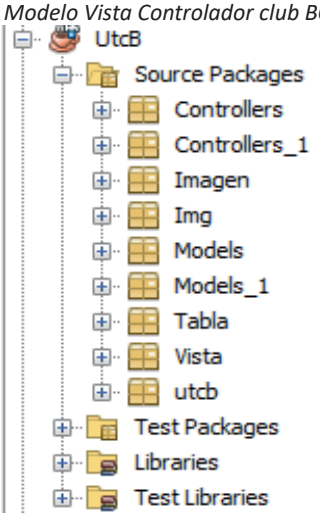
Fig 7.
Diagrama de clases del club de robòtica



propiedades que permiten simplificar el código mediante diferentes vistas, proporciona asistentes de ayuda, para estructurar la visualización de código de manera ordenada. Está centrado en la programación orientada a objetos[10]. Los programadores mediante la manipulación de los objetos creados, adquieren una lógica de programación necesaria para el desarrollo del código.

De tal forma que, NetBeans ha aportado en el proyecto un desarrollo eficaz, ayudando al equipo de programadores a entender el funcionamiento de cada línea de código y de la estructura del mismo, mediante el modelo MVC, Fig 8.

Fig 8
Modelo Vista Controlador club BOTS UTC.



Con la finalidad de agilizar el avance de su desarrollo y cumplir con los requisitos del usuario este sistema cuenta con dos presentaciones una destinada a la vista del administrador y otra para los usuarios normales.

La vista del Administrador, es destinada para realizar el CRUD (Create, Read, Update y Delete) de cada sección antes mencionada. La vista Miembro está destinada para consultas del club de robótica. La aplicación del lenguaje de alto nivel Java.

4. Resultados

Gracias al arduo trabajo y a la planificación detallada, se afirma que, es necesario obtener todos los requerimientos del cliente, los resultados alcanzados se evidencian en el aplicativo funcional que fue implementado, se muestra Fig 9

De acuerdo a las afirmaciones de cada autor de los artículos leídos acerca de las herramientas utilizadas para el desarrollo del software, podemos definir que las herramientas utilizadas para la gestión de procesos del club, son necesarias a comparación de otras, estas ayudan al desempeño óptimo del aplicativo.

Como resultado final del desarrollo del software se ha logrado satisfacer y solventar algunas inconsistencias que se identificaron después de realizar un análisis al modelo de trabajo con el que se maneja actualmente la entidad para la cual se está desarrollando el proyecto, para poder solventar los problemas identificados se decidió diseñar diferentes módulos dentro del aplicativo

cada uno centrado en automatizar y mejorar el proceso de desarrollo de varias actividades como:

Gestión de miembros: En este apartado se pudo mejorar varias subactividades que se realizan dentro de esta actividad entre estos se destaca lo siguiente: se facilita el proceso de gestión de miembros ya que se presenta una interfaz gráfica amigable, intuitiva y fácil de utilizar mejorando así el cumplimiento de esta tarea aquí se podrá agregar, editar y eliminar los datos de cualquier usuario en todo momento que sea necesario. Para acelerar el tedioso proceso de búsqueda de un usuario en específico se ha implementado además la opción de filtrar los registros a partir de los parámetros de búsqueda ingresados destacando que los resultados son presentados automáticamente y al instante, como se ilustra en la Fig.10.

Almacenar documentos: Para la gestión de todos los documentos generados por la entidad se ha diseñado un apartado el cual contará con las siguientes funcionalidades se podrá añadir, modificar y eliminará todos los documentos que se hayan adjuntado , cada uno de ellos podrá ser agregado a una categoría la cual puede ser público o privado finalmente se podrá realizar la búsqueda de un documento específico de una manera rápida y fácil mediante la opción añadida simplemente ingresando los parámetros de búsqueda. Como se presenta en la Fig. 11.

Registro de activos: Para finalizar con las necesidades que se identificó se ha diseñado un apartado dedicado al registro de todos y cada uno de los activos que posee la entidad mediante la implementación del módulo activos el cual cuenta con las siguientes características: se puede añadir, eliminar y modificar todos y cada uno de los registros ingresados además para facilitar la localización de los mismos se puede realizar la búsqueda de un artículo en específico ingresando únicamente los parámetros de búsqueda y se devolverá los resultados inmediatamente. Como se indica en la Fig. 12

Fig 9.

Interfax principal del aplicativo BOTS UTC..

Fig 10.

I*nterfaz miembros del aplicativo BOTS UTC*

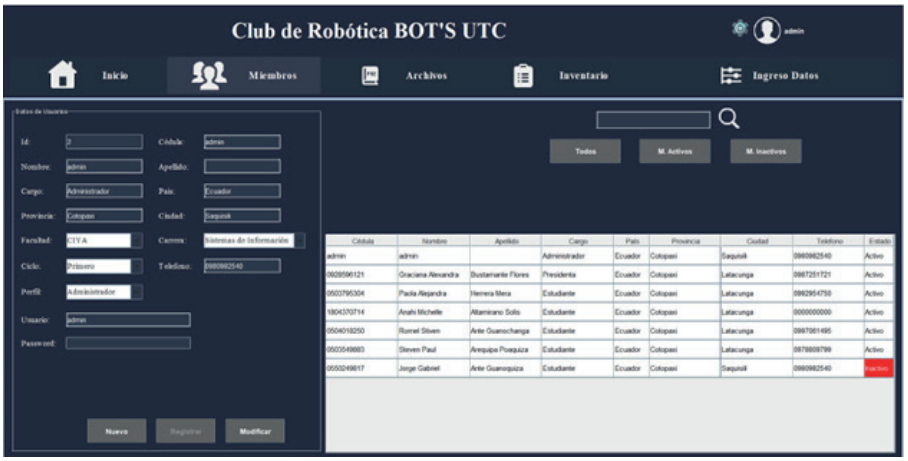


Fig 11.

Interfaz archivos del aplicativo BOTS UTC..

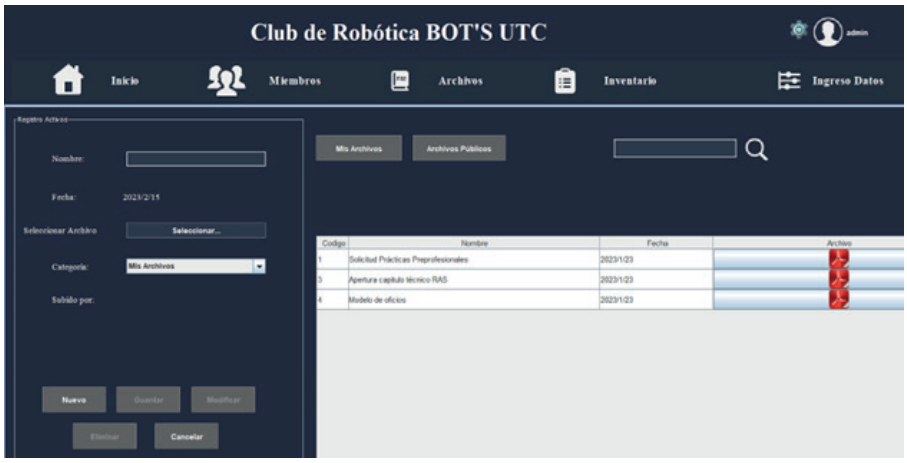
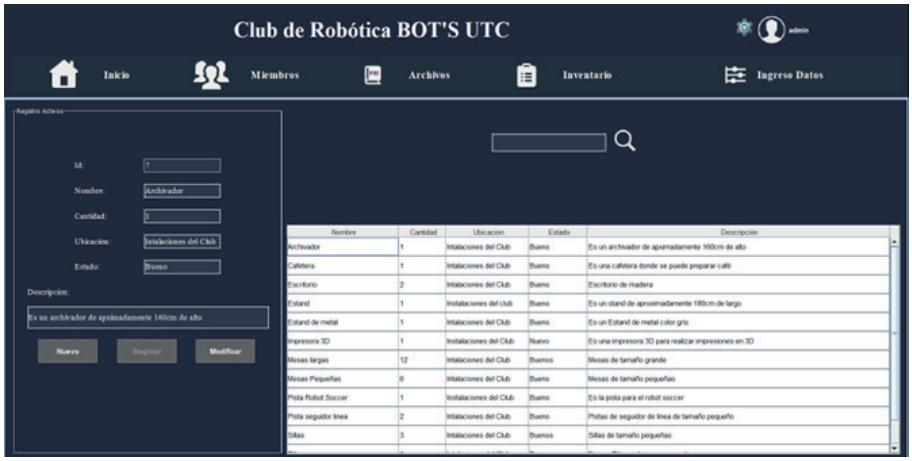


Fig 12.

I*nterfaz activos del aplicativo BOTS UTC*



5. Conclusion

Es necesario elaborar un plan estratégico al inicio del proyecto, para de esta manera cumplir con la entrega del aplicativo. No obstante, estás herramientas fueron necesarias para la ejecución total del proyecto. De acuerdo a la revisión bibliográfica, se denota que el lenguaje Java, permite una programación estructurada gracias a NetBeans, el cual incentiva al programador a optar por modelos que le ayuden de alguna forma a organizar el código para entendimiento del equipo. Es importante destacar que, las herramientas utilizadas en el repositorio de información del club de robótica, son las ideales para trabajar en algún otro desarrollo de software, de acuerdo a la revisión bibliográfica, se puede afirmar que estás agilizan el trabajo del programador. Igualmente, a futuro se sugiere odesarrollar el aplicativo bajo otro enfoque de desarrollo, por ejemplo, orientado al dominio (domain-driven) y realizar comparaciones de ambos enfoques (data-driven y domain-driven). Dada la experiencia de los programadores, se decidió trabajar con el modelo en cascada, sabiendo que el mismo no es flexible en cuanto a cambios posteriores.

Referencias

[1] IBM Documentation. (s.f.-a). IBM - Deutschland | IBM. https://www.ibm.com/docs/es/ida/9.1.2?topic=modeling-logical-data-modelshttps://www.ibm.com/docs/es/ida/9.1.2?topic=modeling-logical-data-models

[2] J. Padhye, V. Firoiu, and D. Towsley, “A stochastic model of TCP Reno congestion avoidance and control,” Univ. of Massachusetts, Amherst, MA, CMPSCI Tech. Rep. 99-02, 1999.

[3] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, IEEE Std. 802.11, 1997.

[4] S. P. Lombas, «Universidad de Valladolid,» 2020. [En línea]. Available: http://uvadoc.uva.es/handle/10324/44428. [Último acceso: 30 Enero 2023].

[5] S. Cabrera Arévalo y Z. Reyes Sánchez, «Desarrollo de un servicio web para el análisis de tendencias de mercado, a través de la extracción de datos de la red social Twitter, mediante la herramienta R, para el apoyo en la toma de decisiones en empresas comerciales,» Universidad de Guayaquil. Facultad de Ciencias Matemáticas y Físicas.Carrera en Ingeniería en Sistemas Computacionales, Guayaquil, 2017.

[6] J. G. T. E. A. Camarena Segrado, M. Martinez Reyes y M. d. L Lopez Garci, «Automatizaciòn de la codificaciòn del patrón

modelo vista controlador (MVC) en proyectos orientados a la Web,» Redalyc.org, vol. 19, nº 3, pp. 239-250, modelo vista controlador (MVC) en proyectos orientados a la Web,» Redalyc.org, vol. 19, nº 3, pp. 239-250, 2012.

[7] IBM Documentation. (s.f.). IBM - Deutschland | IBM.https://www.ibm.com/docs/es/radfws/9.6?topic=SSRTLW_9.6.0/com.ibm.datatools.core.ui.doc/topics/cphysmod.htm

[8] Jonathan Aguilar-Alvarado Ramiro Quezada-Sarmiento Karina García-Galarza, "Aplicación Java para el control de RB Mikrotik en empresas proveedoras de servicio de Internet", en Sistema de Información Científica Redalyc, Red de Revistas Científicas. Machala: UNEMI, 2017, p. 6. Accedido el 31 de enero de 2023. [En línea]. Disponible: https://www.redalyc.org/journal/5826/582661257015/

[9] Calendamaia. "NetBeans". Genbeta - Software, descargas, aplicaciones web y móvil, desarrollo. https://www.genbeta.com/desarrollo/netbeans-1#:~:text=Netbeans%20es%20un%20entorno%20de,Web,%20o%20para%20dispositivos%20móviles. (accedido el 31 de enero de 2023).

[10] D. A. A. R. I. B. Flor Eugenia Narciso Farias, «Sistema Web para la Gestion de Indicadores del CAU de RedULA,» de Primera Conferencia Nacional de Computaciòn, Informática y Sistemas (CoNCISa 2013), Naiguatá, 2013.

[11] N. Ford, R. Parsons y P. Kua, Building Evolutionary Architectures, Sebastopol: O'Reilly Media, Inc., 2017.

[12] M. Navarro, M. Moreno, J. Aranda, L. Parra y J. Rueda, «Arquitectura de software en el proceso de desarrollo ágil una perspectiva basada en requisitos significantes para la arquitectura,» *XX Workshop de Investigadores en Ciencias de la Computación,* pp. 635-639, 2018.

[13] A. Hernández Paez, J. A. Domínguez Falcón y A. A. Pi Cruz, «Arquitectura de software para el desarrollo de videojuegos sobre el motor de juego Unity 3D,» *Revista de I+D Tecnológico,* vol. 14, nº 1, pp. 54-65, 2018.

[14] J. Nieto Sánchez, «Modelo de Arquitectura de Software para Aplicaciones iOS basado en Clean Architecture,» de *Facultad de Matemáticas e Informática Universidad de Barcelona*, Barcelona, 2017.

[15] O. Blancarte, «Reactiveprogramming,» 2020. [En línea]. Available: https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/capas. [Último acceso: 19 Enero 2022].

[16] J. Peláez, «Geeks,» Patterns and Practices de Microsoft., Mayo 2009. [En línea]. Available: https://geeks.ms/jkpelaez/2009/05/30/arquitectura-basada-en-capas/. [Último acceso: 19 Enero 2022].

[17] A. Schiaffarino, «Infranetworking,» 12 Marzo 2019. [En línea]. Available: https://blog.infranetworking.com/modelo-cliente-servidor/. [Último acceso: 20 Enero 2022].

[18] S. Peltonen, L. Mezzalira, and D. Taibi, “Motivations, benefits, and issues for adopting Micro-Frontends: A Multivocal Literature Review,” *Inf. Softw. Technol.*, vol. 136, no. July 2020, 2021, doi: 10.1016/j.infsof.2021.106571Gf

[19] P. Oukes, M. van Andel, E. Folmer, R. Bennett, and C. Lemmen, “Domain-Driven Design applied to land administration system development: Lessons from the Netherlands,” *Land use policy*, vol. 104, no. July 2020, p. 105379, 2021, doi: 10.1016/j.landusepol.2021.105379.

[20] K. Peguero and X. Cheng, “CSRF protection in JavaScript frameworks and the security of JavaScript applications,” *High-Confidence Comput.*, vol. 1, no. 2, p. 100035, 2021, doi: 10.1016/j.hcc.2021.100035.

[21] R. R. Panda and N. K. Nagwani, “Classification and intuitionistic fuzzy set based software bug triaging techniques,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 8, pp. 6303–6323, 2022, doi: 10.1016/j.jksuci.2022.01.020.

[22] A. U. Rehman, Y. Khan, R. U. Ahmed, N. Ullah, and M. A. Butt, “Human tracking robotic camera based on image processing for live streaming of conferences and seminars,” *Heliyon*, vol. 9, no. 8, p. e18547, 2023, doi: 10.1016/j.heliyon.2023.e18547.

[23] A. U. Rehman, Y. Khan, R. U. Ahmed, N. Ullah, and M. A. Butt, “Human tracking robotic camera based on image processing for live streaming of conferences and seminars,” *Heliyon*, vol. 9, no. 8, p. e18547, 2023, doi: 10.1016/j.heliyon.2023.e18547.