PROGRAMMASY
1.0
06-03-2024

Documento de Especificación de Arquitectura

Realizado por:

Ruben Duran
Jorge Hernan Agudelo
Cristian Camilo Mesa

HISTORIAL DE REVISIONES

Fecha	Versión	Autor	Descripción	Revisado Por
06-03-2024	1.0.0	Equipo de Desarrollo	Informe DEA IEEE	Mary Rubiano

PROGRAMMASY
1.0
06-03-2024

Contenido

1. Documento de Arquitectura de Software

1.1. Introducción

El presente documento tiene como finalidad ser un informe de arquitectura de software del proyecto integrador denominado Programmasy. El proyecto es un programa web de auto-formación en el mundo de la programación, las personas interesadas en aumentar sus conocimientos en lenguajes de programación, servicios web, bases de datos, entre otros.

1.2. Propósito

Nuestra es lograr brindarles a las personas la posibilidad de tener a la mano y de simple acceso a recursos que lo guíen, le impulsen o le integre los conocimientos necesarios para iniciar, reforzar o mejorar sus habilidades en el mundo de la programación.

1.3. Alcance

Queremos llegar a todas las personas que necesiten o requieran una herramienta de fácil uso para poder formarse como programador o desarrollador, ya que en este mundo los niveles de programadores van de Junior, Semi-senior y Senior, aseguramos que nuestro amplio paquete de recursos que van desde cursos, videos o documentos garantizan la posibilidad de incrementar y reforzar las habilidades de cualquier programador o desarrollador.

1.4. Referencias

1. Documento de Especificación de Requerimientos no funcionales.

Requerimientos No Funcionales.

Identificación del requerimiento:	RNF1
Nombre del Requerimiento:	Interfaz del sistema.
Características:	El sistema presentara una interfaz de usuario sencilla para que sea de fácil manejo a los usuarios del sistema.
Descripción del requerimiento:	El sistema debe tener una interfaz de uso intuitiva y sencilla.
Prioridad del requerimiento: Alta	

Identificación del requerimiento:	RNF2
Nombre del Requerimiento:	Ayuda en el uso del sitema.
Características:	La interfaz del usuario deberá de presentar un sistema de ayuda para que los mismos usuarios del sistema se les faciliten el trabajo en cuanto al manejo del sistema.
Descripción del requerimiento:	La interfaz debe estar complementada con un buen sistema de ayuda (la administración puede recaer en personal con poca experiencia en el uso de aplicaciones informáticas).
Prioridad del requerimiento: Alta	

Identificación del requerimiento:	RNF3
Nombre del Requerimiento:	Mantenimiento.

2. Visión del Proyecto.

   El estado del proyecto está en curso, aunque en visualización a futuro queremos entregar un producto completo el cual sea bien recibido y muy utilizado y recomendado, la razón del proyecto en esencia es poder entrar al mercado de los programadores y poder influir en el aprendizaje que estos mismos quieren obtener.

1.5. Definiciones acrónimos y abreviaciones

ARQUITECTURA DE SOFTWARE: Para nuestro proyecto, elegimos utilizar una arquitectura de software que se acopla a las necesidades del mismo, ya que este es un programa web, la arquitectura de micro servicios establece en nivel de funcionabilidad y estructura una visualización y mecanismo moderno a las percepciones que engloban los sistemas basados en la web.

DESCRIPCION DE ARQUITECTURA: La arquitectura de micro servicios es un enfoque moderno y altamente escalable para el desarrollo de aplicaciones. En lugar de construir una sola aplicación monolítica, se divide en componentes pequeños e independientes llamados micro servicios. Cada micro servicio cumple una función específica y se comunica con otros a través de APIs. Aquí están algunas características clave de esta arquitectura:

Componentes Independientes: Los micro servicios se desarrollan, implementan y operan de forma independiente. Esto permite una implementación rápida y fácil de cada función de la aplicación.

Fácil Mantenimiento y Pruebas: Los equipos pueden experimentar con nuevas funciones y revertirlas si no funcionan. Además, facilita el aislamiento y corrección de errores en servicios individuales.

Equipos Pequeños y Multifuncionales: Los equipos crean y mantienen micro servicios, lo que fomenta prácticas ágiles y DevOps. Pueden trabajar de forma independiente y moverse rápidamente.

Organización en Torno a Capacidades Empresariales: Los servicios se agrupan según las capacidades empresariales. Los equipos trabajan para crear una funcionalidad específica.

Infraestructura Automatizada: Los equipos utilizan prácticas de automatización de infraestructuras, como CI/CD, para crear e implementar servicios sin afectar a otros equipos.

VISTAS:

Aplicación de Comercio Electrónico
---

1. Servicio de Usuarios
   - Maneja la información del usuario, como detalles de la cuenta y preferencias de usuario.

2. Servicio de Productos
   - Gestiona el catálogo de productos, incluyendo detalles del producto y disponibilidad de stock.

3. Servicio de Carrito de Compras
   - Mantiene el estado del carrito de compras del usuario y permite agregar o eliminar productos del carrito.

4. Servicio de Pedidos
   - Maneja la creación de pedidos, el seguimiento del estado del pedido y la notificación al usuario.

5. Servicio de Pago
   - Procesa los pagos y se comunica con los proveedores de servicios de pago externos.

Cada uno de estos servicios puede desarrollarse, implementarse y escalar de forma independiente. Cada servicio tiene su propia base de datos y se comunica con los otros servicios a través de API.

Esta es solo una representación simplificada. En una aplicación real, cada servicio puede tener subcomponentes y puede haber servicios adicionales para funciones como búsqueda, recomendaciones, logística, etc. Además, también habría consideraciones adicionales para cosas como el manejo de errores, la seguridad y la tolerancia a fallos.

STAKEHOLDER:

1. **Estudiantes o Usuarios**:
   - Son los principales beneficiarios del programa. Su éxito en el aprendizaje es fundamental.
   - Tienen interés en la calidad del contenido, la accesibilidad, la interactividad y la efectividad del programa.
2. **Desarrolladores o Instructores**:
   - Crean y mantienen el contenido del programa.
   - Están interesados en la claridad, precisión y relevancia del material, así como en la facilidad de actualización.
3. **Organizaciones Educativas o Instituciones**:
   - Pueden ser universidades, escuelas o empresas que ofrecen el programa.
   - Tienen un interés en la reputación, el éxito académico y la retención de estudiantes.
4. **Empresas Tecnológicas o Empleadores**:
   - Buscan candidatos con habilidades en lenguajes de programación.
   - Pueden colaborar con el programa para adaptar el contenido a las necesidades del mercado laboral.
5. **Proveedores de Plataformas o Tecnología**:
   - Ofrecen la infraestructura tecnológica para el programa (por ejemplo, servidores, bases de datos, plataformas de aprendizaje en línea).
   - Les interesa la eficiencia, escalabilidad y seguridad del programa.

6. **Comunidad de Desarrolladores**:
   - Pueden contribuir al programa mediante comentarios, correcciones o sugerencias.
   - Les preocupa la calidad y relevancia del contenido.
7. **Patrocinadores o Financiadores**:
   - Proporcionan recursos financieros para el desarrollo y mantenimiento del programa.
   - Tienen un interés en el éxito general del programa.
8. **Gobiernos o Reguladores**:
   - Pueden establecer políticas o estándares relacionados con la educación en línea.
   - Les preocupa la legalidad, la privacidad y la equidad.
9. **Comunidad Académica o Investigadores**:
   - Pueden estudiar el impacto del programa en el aprendizaje y la retención.
   - Les interesa la innovación pedagógica y la mejora continua.
10. **Usuarios Finales o Clientes**:
    - Si el programa tiene un componente comercial (por ejemplo, suscripciones pagas), los usuarios finales son stakeholders importantes.
    - Les preocupa la calidad del servicio y la satisfacción del cliente.

2. Generalidades del Proyecto

2.1. Problema a Resolver

El problema a resolver del proyecto es básicamente mantener en constante acercamiento al usuario que utilice nuestro programa, es decir, nos enfrentamos a la problemática de que los usuarios tengan el incentivo y la motivación constante de hacer uso del aplicativo y de las diferentes herramientas del mismo.

2.2. Descripción General del Sistema a Desarrollar

Las áreas o módulos que se deben desarrollar para el programa son dos, el primer módulo consta del sitio general en el que los usuarios estarán alojados, este sitio tendrá como contenido los recursos que los usuarios podrán utilizar, además de que también estará el panel del perfil del usuario en donde se le mostrará sus habilidades, su nombre, su progreso, su nivel y también la posibilidad de la edición de los datos del usuario. El segundo módulo será el chat de los usuarios, el cual tendrá un panel del perfil del usuario donde se visualizarán los grupos los cuales el pertenece y también se hará la visualización de las solicitudes de mensajes que le lleguen a su chat.

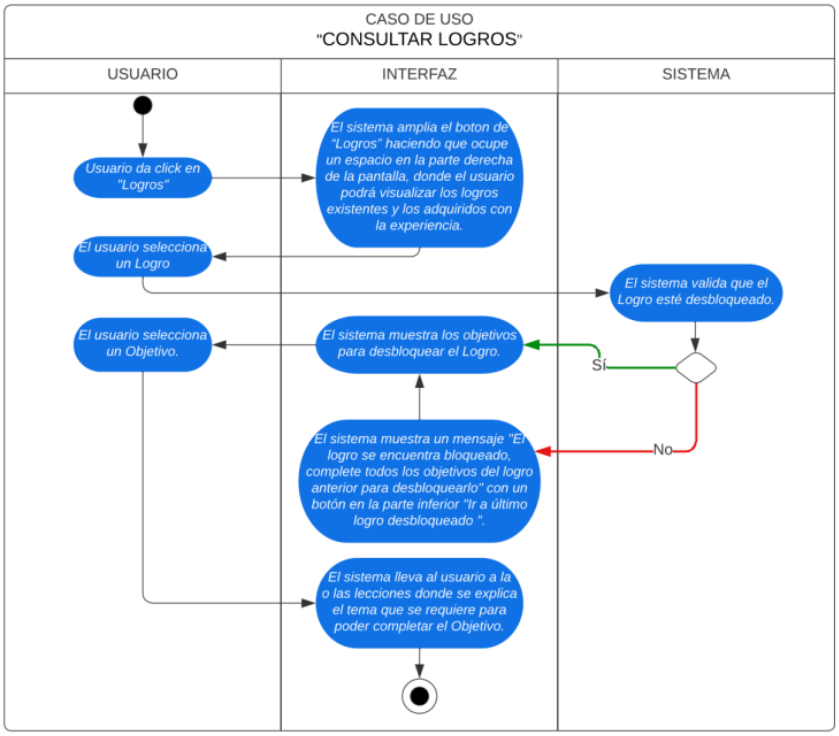
2.3. Identificación de los Stakeholders y sus responsabilidades

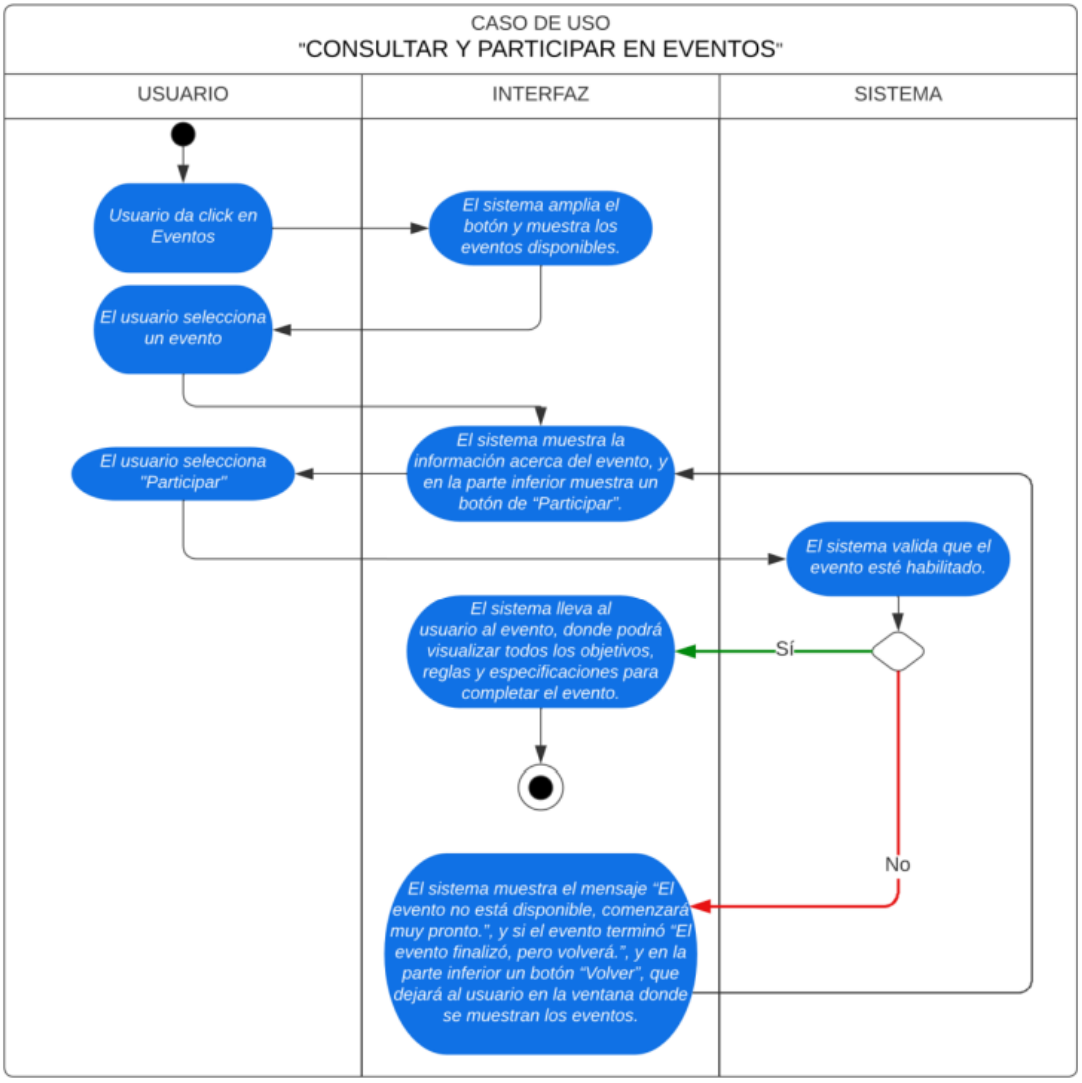
STAKEHOLDER	DESCRIPCIÓN	ESCENARIO	Caso de Uso
Administrador	Es el perfil de usuario que administra y maneja el aplicativo.	- Escenario de negocios	- Gestionar el contenido de los recursos - Administrar a los usuarios reportados o que estén eliminados
Usuario	Es el perfil de usuario el cual hará uso de las herramientas del programa y que interactuara con esta misma.	- Escenario de diseño	- Inicio de sesión - Registro - Consumo de los recursos - Participación en el chat global y en los foros

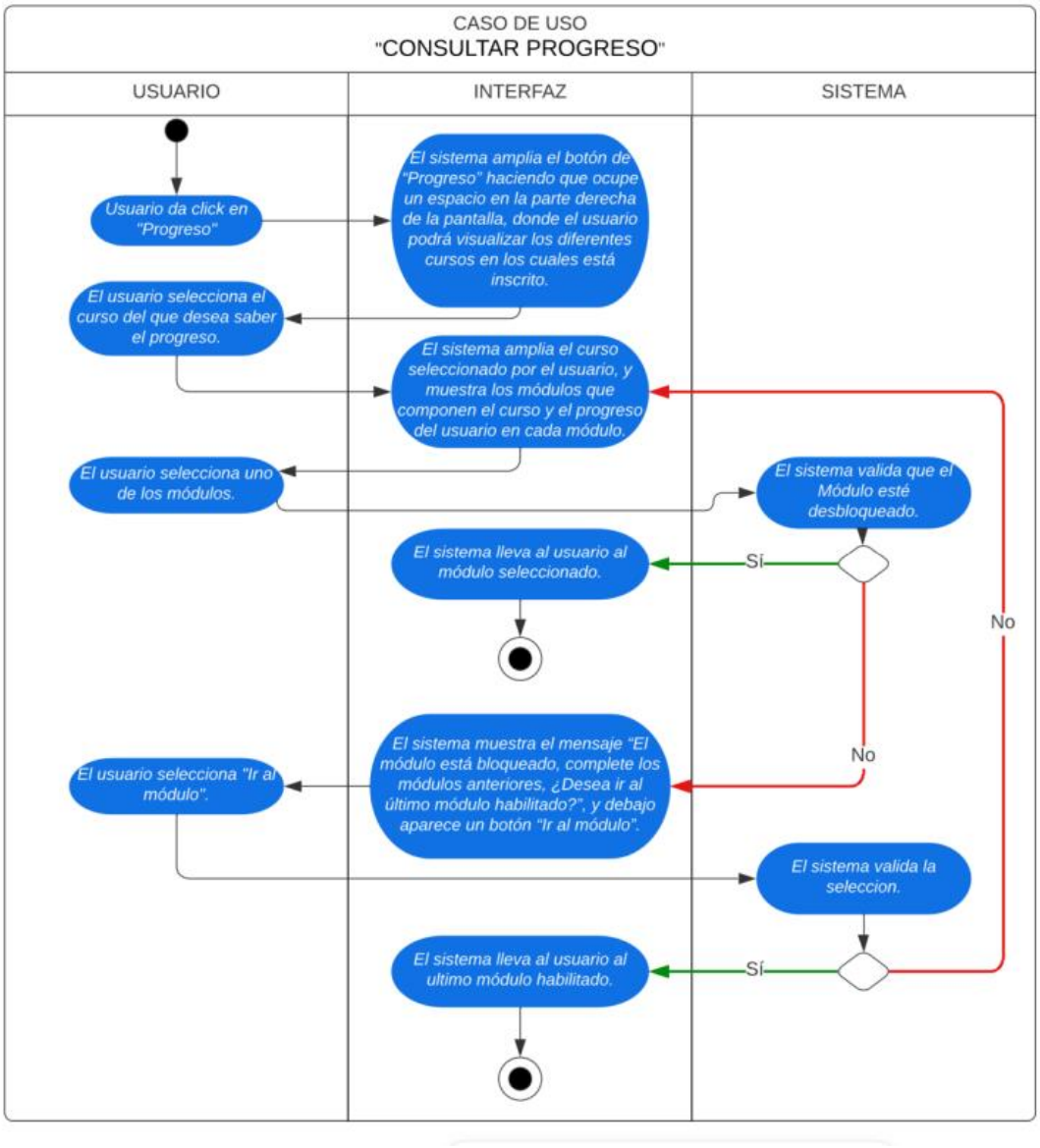
3. Vistas de la arquitectura

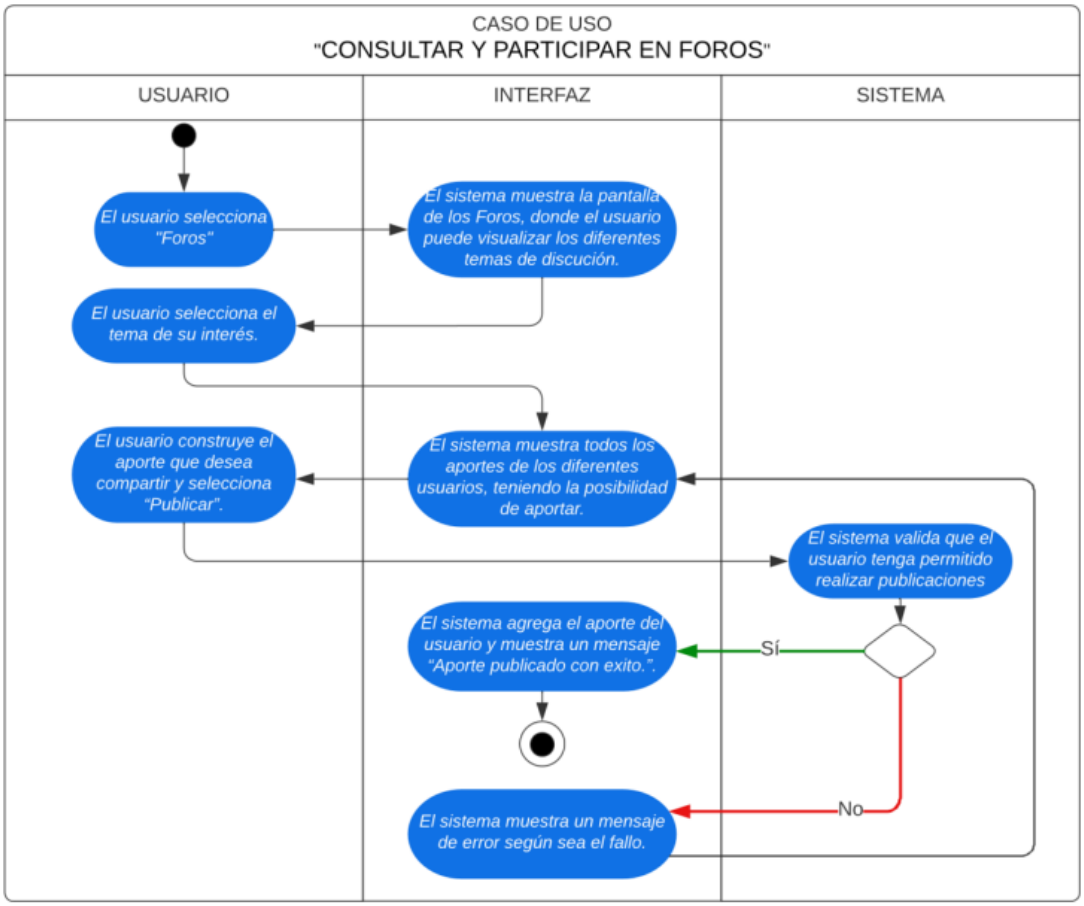
3.1. Vista de Casos de Uso

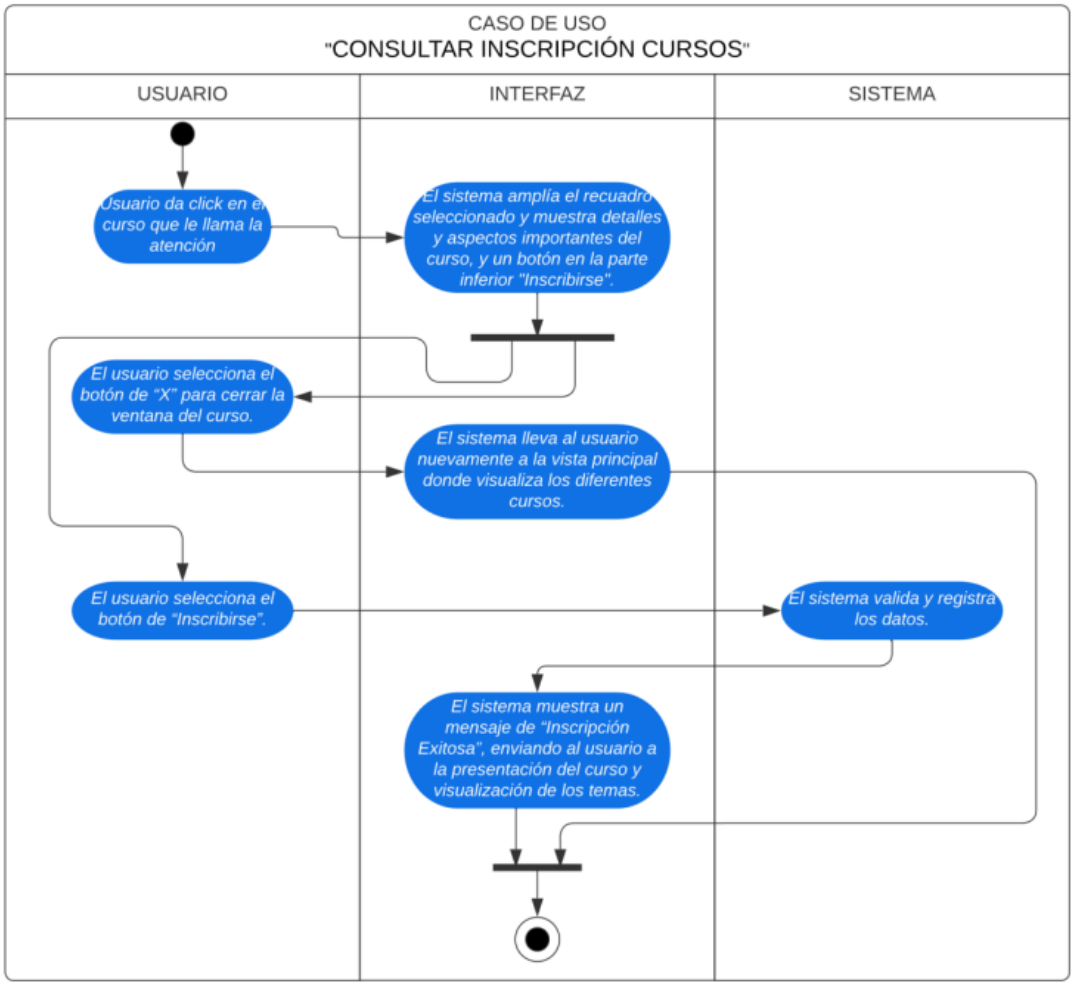
1- CONSULTAR LOGROS

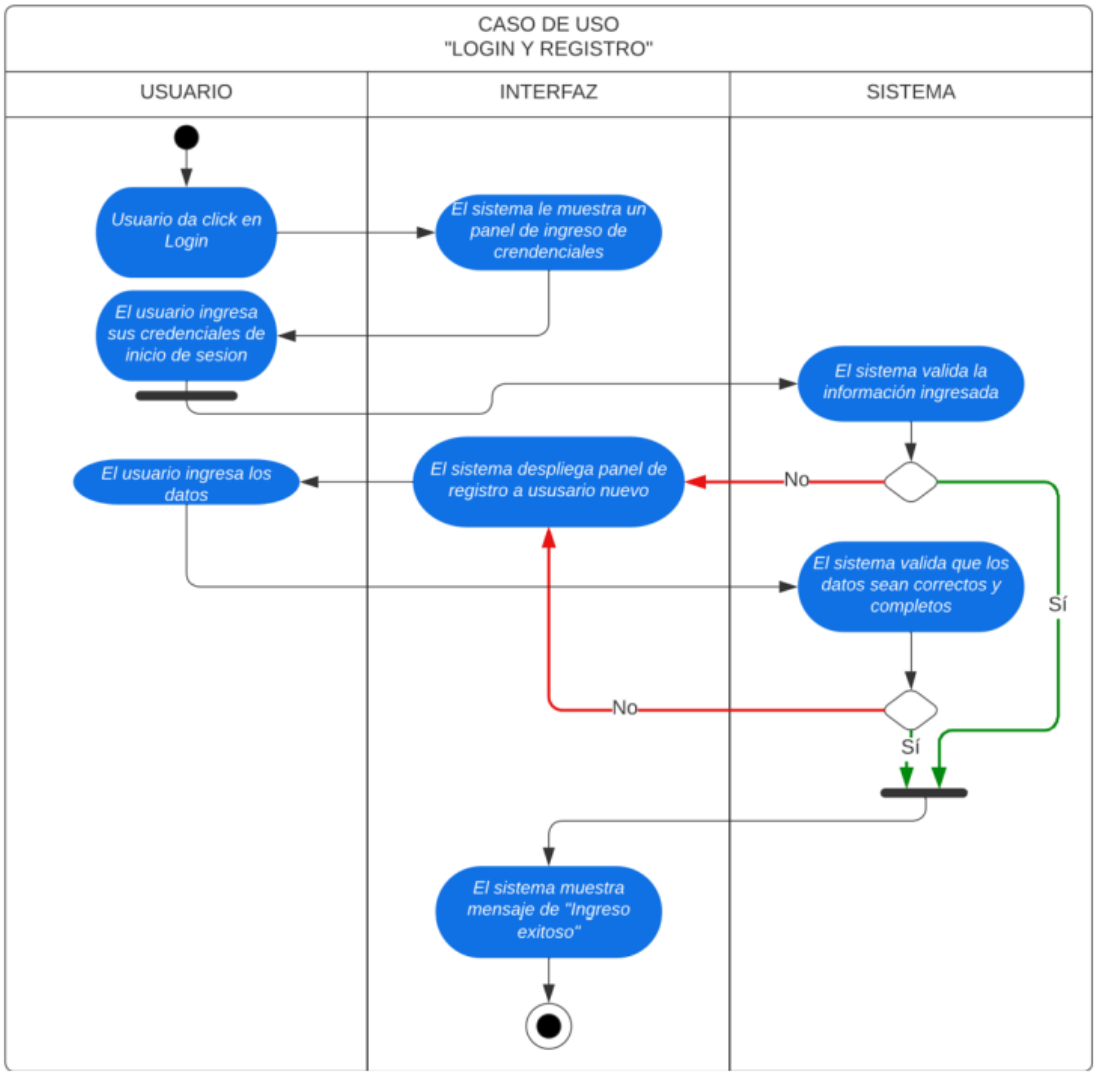



CASO DE USO
"CONSULTAR Y PARTICIPAR EN EVENTOS"
USUARIO
INTERFAZ
SISTEMA
Usuario da click en Eventos
El sistema amplia el botón y muestra los eventos disponibles.
El usuario selecciona un evento
El sistema muestra la información acerca del evento, y en la parte inferior muestra un botón de "Participar".
El usuario selecciona "Participar"
El sistema valida que el evento esté habilitado.
Sí
El sistema lleva al usuario al evento, donde podrá visualizar todos los objetivos, reglas y especificaciones para completar el evento.
No
El sistema muestra el mensaje "El evento no está disponible, comenzará muy pronto.", y si el evento terminó "El evento finalizó, pero volverá.", y en la parte inferior un botón "Volver", que dejará al usuario en la ventana donde se muestran los eventos.


CASO DE USO
"CONSULTAR PROGRESO"
USUARIO
INTERFAZ
SISTEMA
Usuario da click en "Progreso"
El sistema amplia el botón de "Progreso" haciendo que ocupe un espacio en la parte derecha de la pantalla, donde el usuario podrá visualizar los diferentes cursos en los cuales está inscrito.
El usuario selecciona el curso del que desea saber el progreso.
El sistema amplia el curso seleccionado por el usuario, y muestra los módulos que componen el curso y el progreso del usuario en cada módulo.
El usuario selecciona uno de los módulos.
El sistema valida que el Módulo esté desbloqueado.
Sí
El sistema lleva al usuario al módulo seleccionado.
No
No
El sistema muestra el mensaje "El módulo está bloqueado, complete los módulos anteriores, ¿Desea ir al último módulo habilitado?", y debajo aparece un botón "Ir al módulo".
El usuario selecciona "Ir al módulo".
El sistema valida la seleccion.
Sí
El sistema lleva al usuario al ultimo módulo habilitado.

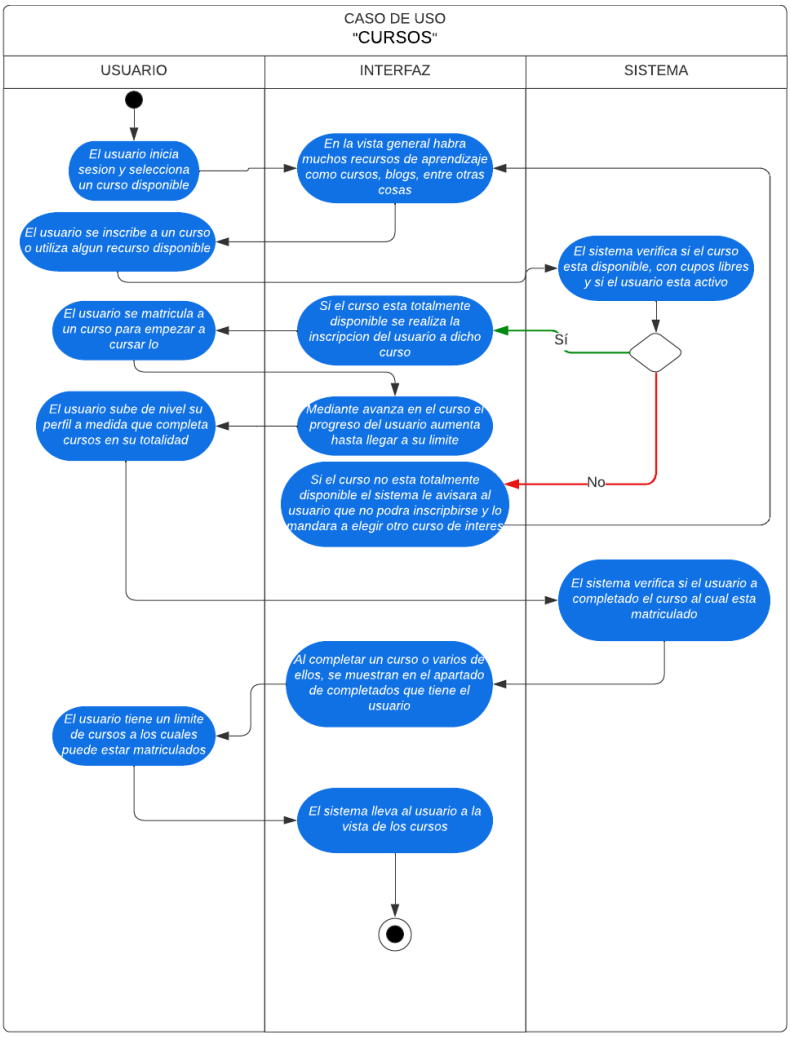

CASO DE USO
"CONSULTAR Y PARTICIPAR EN FOROS"
USUARIO
INTERFAZ
SISTEMA
El usuario selecciona "Foros"
El sistema muestra la pantalla de los Foros, donde el usuario puede visualizar los diferentes temas de discución.
El usuario selecciona el tema de su interés.
El usuario construye el aporte que desea compartir y selecciona "Publicar".
El sistema muestra todos los aportes de los diferentes usuarios, teniendo la posibilidad de aportar.
El sistema valida que el usuario tenga permitido realizar publicaciones
Sí
El sistema agrega el aporte del usuario y muestra un mensaje "Aporte publicado con exito.".
No
El sistema muestra un mensaje de error según sea el fallo.


CASO DE USO
"CONSULTAR INSCRIPCIÓN CURSOS"
USUARIO
INTERFAZ
SISTEMA
Usuario da click en el curso que le llama la atención
El sistema amplía el recuadro seleccionado y muestra detalles y aspectos importantes del curso, y un botón en la parte inferior "Inscribirse".
El usuario selecciona el botón de "X" para cerrar la ventana del curso.
El sistema lleva al usuario nuevamente a la vista principal donde visualiza los diferentes cursos.
El usuario selecciona el botón de "Inscribirse".
El sistema valida y registra los datos.
El sistema muestra un mensaje de "Inscripción Exitosa", enviando al usuario a la presentación del curso y visualización de los temas.

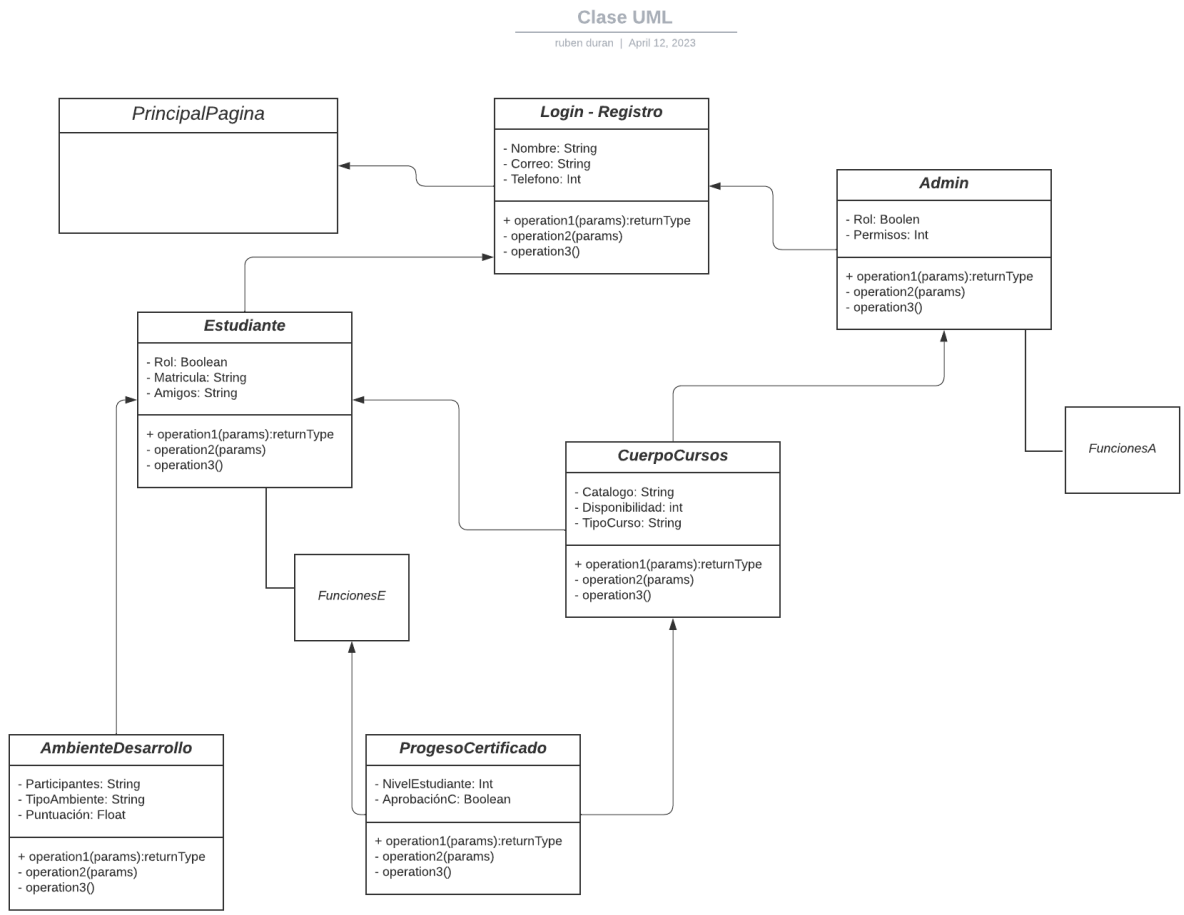

CASO DE USO
"LOGIN Y REGISTRO"
USUARIO
INTERFAZ
SISTEMA
Usuario da click en Login
El sistema le muestra un panel de ingreso de crendenciales
El usuario ingresa sus credenciales de inicio de sesion
El sistema valida la información ingresada
El usuario ingresa los datos
El sistema despliega panel de registro a ususario nuevo
No
El sistema valida que los datos sean correctos y completos
Sí
No
Sí
El sistema muestra mensaje de "Ingreso exitoso"

3.2. Vista de Procesos

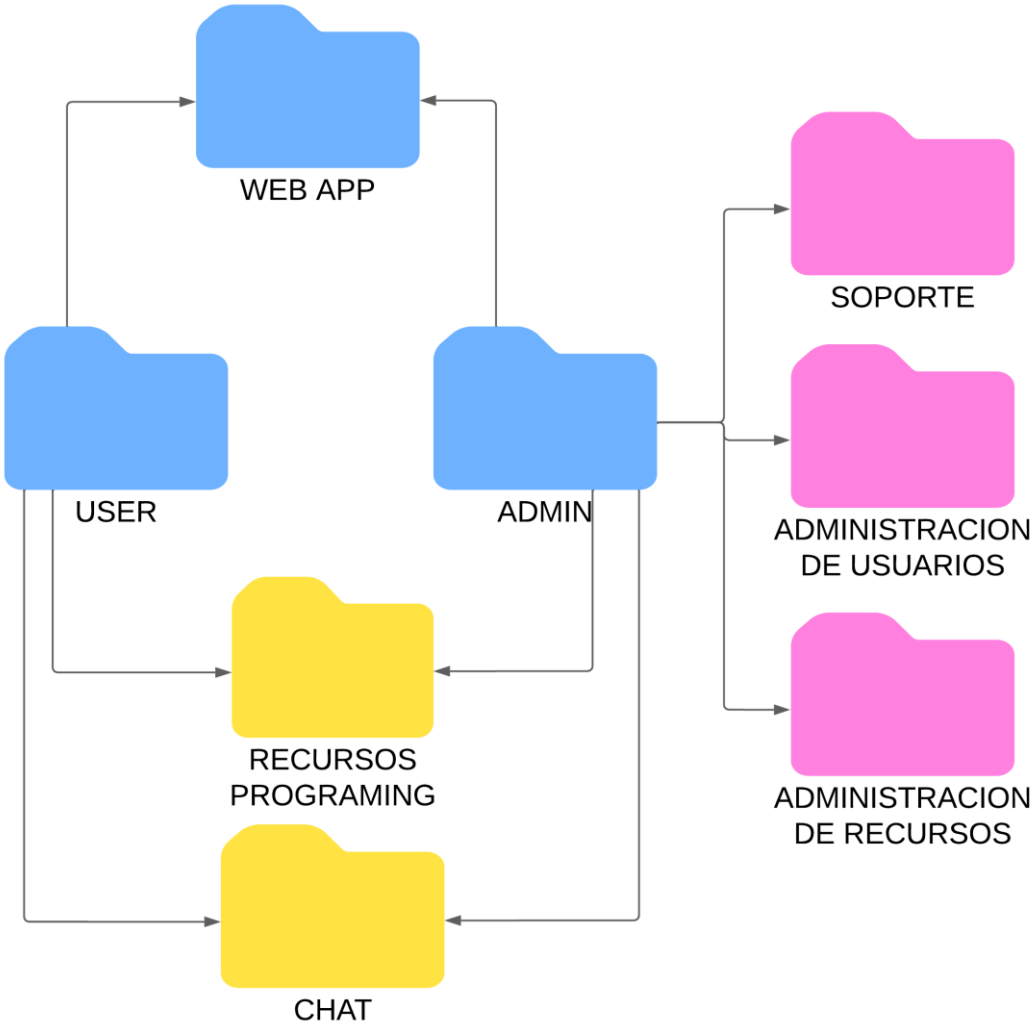
3.2.1. Diagrama de Actividades



3.3. Vista Lógica
3.3.1. Diagramas – Clases

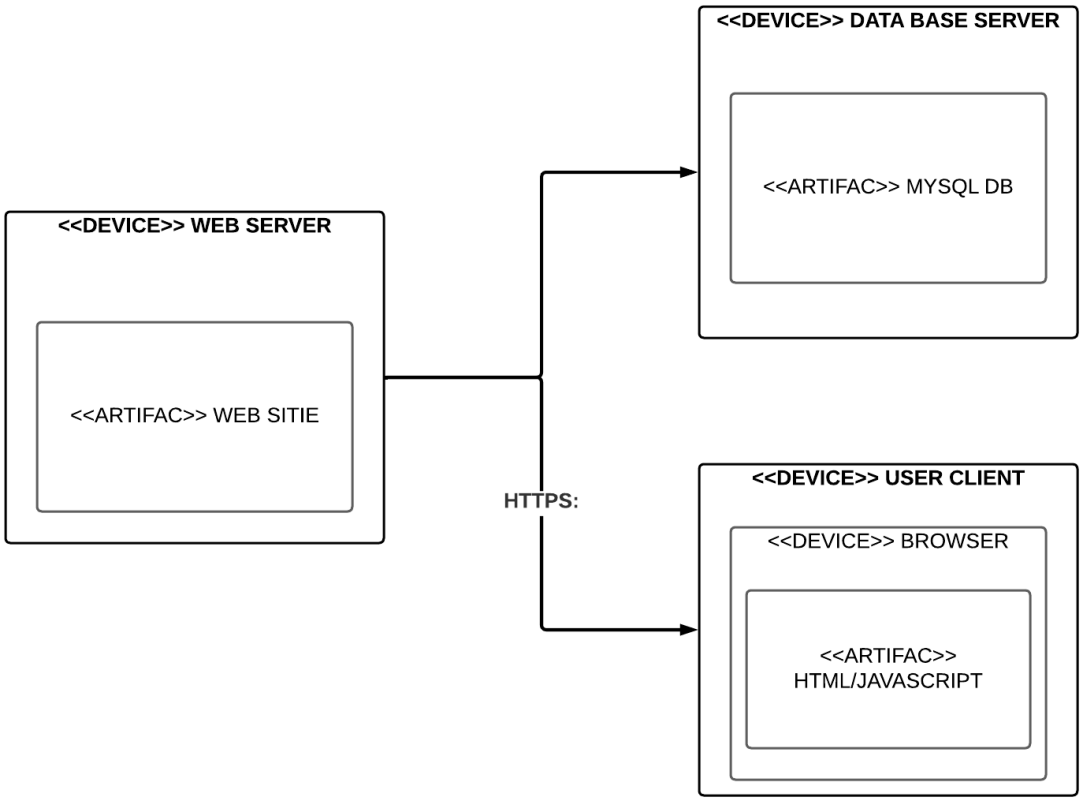


3.4. Vista de Implementación
3.4.1. Diagrama de Paquetes



3.5. Vista de Despliegue

3.5.1. Diagrama de despliegue



1. Servidor de Aplicaciones
   - Servicio de Autenticación: Maneja el inicio de sesión y la autenticación de los usuarios.
   - Servicio de Usuarios: Gestiona la información del perfil del usuario.
   - Servicio de Cursos: Maneja la información de los cursos disponibles y el progreso del usuario.
   - Servicio de Evaluación: Genera y califica las pruebas de los usuarios.

2. Base de Datos
   - Base de Datos de Usuarios: Almacena la información del perfil del usuario.
   - Base de Datos de Cursos: Almacena la información de los cursos y el progreso del usuario.
   - Base de Datos de Evaluaciones: Almacena las pruebas generadas y las respuestas de los usuarios.

3. Servidor Web
   - Servidor de Contenido Estático: Sirve el contenido estático como HTML, CSS y JavaScript.
   - Servidor de Contenido Dinámico: Sirve el contenido dinámico basado en las solicitudes del usuario.

4. Cliente
   - Navegador Web: El usuario interactúa con la aplicación a través del navegador web.

Cada uno de estos componentes puede desarrollarse, implementarse y escalar de forma independiente. Los componentes se comunican entre sí a través de API.

4. Arquitectura en capas

(capas, patrones, plataforma)

La arquitectura de microservicios se divide en varias capas, cada una con su propia responsabilidad:

1. **Capa de Presentación**: Esta es la interfaz de usuario donde los usuarios interactúan con la aplicación. En nuestro caso, es el navegador web del cliente.
2. **Capa de Aplicación**: Esta capa contiene los microservicios, cada uno de los cuales es responsable de una funcionalidad específica (por ejemplo, autenticación, gestión de usuarios, gestión de cursos, evaluación).
3. **Capa de Datos**: Cada microservicio tiene su propia base de datos para garantizar la independencia y evitar el acoplamiento estrecho.

Los patrones de microservicios son estrategias o soluciones comunes para problemas recurrentes en la arquitectura de microservicios. Algunos de los patrones utilizados en nuestro programa web podrían ser:

1. **API Gateway**: Un punto de entrada único para todas las solicitudes de los clientes. El API Gateway enruta las solicitudes al microservicio correspondiente.
2. **Descubrimiento de Servicios**: Permite que los microservicios descubran y se comuniquen entre sí.
3. **Patrón de Circuit Breaker**: Maneja fallos en los microservicios para evitar que se propaguen a otros servicios.

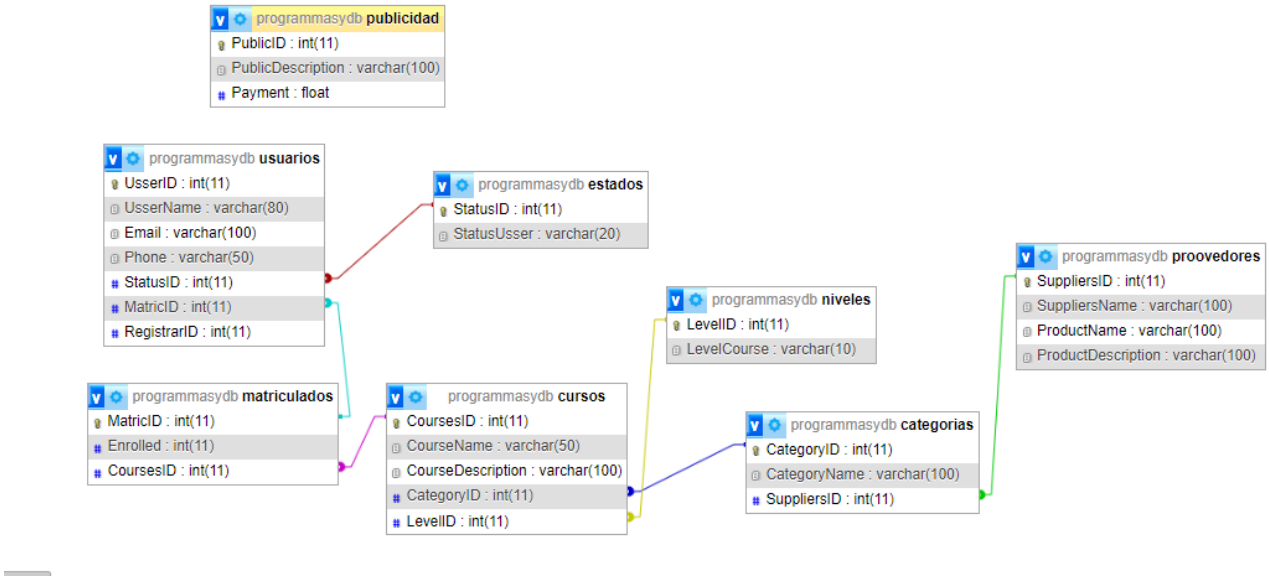
La plataforma es el entorno en el que se implementan y operan los microservicios. En nuestro caso, podríamos utilizar una plataforma como Kubernetes, que proporciona características como:

1. **Orquestación de contenedores**: Permite implementar y gestionar los microservicios en contenedores.
2. **Escalado automático**: Permite escalar los microservicios de forma automática en función de la demanda.

3. **Balanceo de carga**: Distribuye las solicitudes de los clientes entre las instancias de los microservicios.
4. **Gestión de fallos**: Proporciona mecanismos para manejar fallos y garantizar la alta disponibilidad de los microservicios.

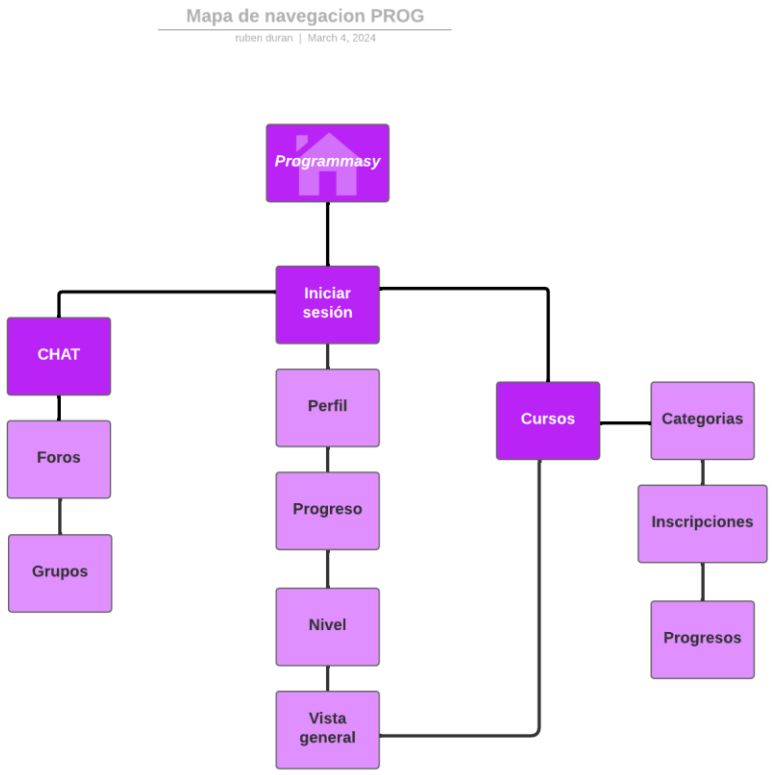
5. Vista de Datos

5.1. Modelo Relacional



6. Definición de Interfaces de Usuario

Mapa de navegación. Demostración de las interfaces



7. Características Generales de Calidad

Para nuestro proyecto necesitamos tener una gestión de calidad que acompañe en los distintos ámbitos dentro del programa, esto incluye desde organización hasta calidad del servicio al usuario, por ende, para programmasy decidimos elegir las siguientes características:

1. **Interfaz de usuario intuitiva**: La interfaz debe ser fácil de usar y navegar para que los usuarios puedan concentrarse en aprender el lenguaje de programación.
2. **Contenido de alta calidad**: El contenido debe ser preciso, actualizado y relevante para el lenguaje de programación que se está aprendiendo.
3. **Ejercicios prácticos**: Debe haber oportunidades para que los usuarios practiquen lo que han aprendido a través de ejercicios y proyectos.
4. **Retroalimentación instantánea**: Los usuarios deben recibir retroalimentación inmediata sobre su código para facilitar el aprendizaje.
5. **Adaptabilidad**: El programa debe adaptarse al nivel de habilidad del usuario y ofrecer contenido personalizado basado en su progreso.
6. **Accesibilidad**: El programa debe ser accesible desde diferentes dispositivos y navegadores para facilitar el aprendizaje en cualquier momento y lugar.

7. **Comunidad de apoyo**: Debe haber una comunidad activa donde los usuarios puedan hacer preguntas y obtener ayuda de otros usuarios.
8. **Recursos adicionales**: Debe proporcionar enlaces a recursos adicionales, como documentación oficial, tutoriales y foros de discusión.
9. **Seguimiento del progreso**: Los usuarios deben poder seguir su progreso y ver cuánto han aprendido.
10. **Seguridad**: El programa debe garantizar la seguridad de los datos del usuario y cumplir con las leyes de privacidad.

8. Stack Tecnológico

Para el desarrollo del programa las herramientas a utilizar e implementar son las siguientes:

Lenguajes de programación a utilizar: para este apartado tanto en back y fornt-end los lenguajes que se implementarán serán JavaScript, HTML y CSS.

JavaScript tiene sus pros y contras al momento de implementarse, por ejemplo:

Pros:

- **Universalidad**: JavaScript es el único lenguaje que se ejecuta nativamente en el navegador, lo que significa que puedes usar el mismo lenguaje en el servidor y en el cliente. Esto puede hacer que el desarrollo sea más eficiente y coherente.
- **Node.js**: Node.js es un entorno de ejecución de JavaScript en el servidor que es rápido y eficiente. Tiene un ecosistema de paquetes muy grande (npm), lo que significa que hay módulos para casi cualquier cosa que quieras hacer.
- **Desarrollo en tiempo real**: JavaScript es ideal para aplicaciones en tiempo real (como chats o juegos) debido a su orientación a eventos y su capacidad para realizar operaciones de E/S sin bloqueo.
- **JSON**: JavaScript Object Notation (JSON) es un formato de intercambio de datos ligero que es fácil de leer y escribir. Como JavaScript maneja JSON de forma nativa, trabajar con APIs REST es muy sencillo.

Contras:

- **Callback Hell**: JavaScript depende en gran medida de las funciones de callback, lo que puede llevar a un fenómeno conocido como “callback hell” si no se maneja correctamente. Sin embargo, esto se ha mitigado en gran medida con la introducción de Promesas y async/await en versiones más recientes de JavaScript.

- **Tipado dinámico**: JavaScript es un lenguaje de tipado dinámico, lo que significa que el tipo de una variable puede cambiar en tiempo de ejecución. Esto puede llevar a errores sutiles y difíciles de detectar.
- **Madurez**: Aunque Node.js ha existido durante más de una década, todavía se considera menos maduro que otros entornos de back-end como Ruby on Rails o Django.

En el apartado de las bases de datos, elegimos Mysql como motor de bases de datos, esta elección se da por las siguientes razones:

- **Ampliamente utilizado**: MySQL es uno de los sistemas de gestión de bases de datos más utilizados en el mundo, lo que significa que hay una gran cantidad de documentación, tutoriales y ejemplos disponibles.
- **Open Source**: MySQL es un software de código abierto, lo que significa que es gratuito para usar y modificar. Esto puede reducir los costos de licencia.
- **Rendimiento**: MySQL es conocido por su velocidad y eficiencia. Puede manejar una gran cantidad de datos y es muy rápido en la recuperación de datos.
- **Seguridad**: MySQL tiene una sólida seguridad incorporada para proteger los datos. Incluye el cifrado de datos, el bloqueo de usuarios y privilegios a nivel de host y de base de datos.
- **Compatibilidad**: MySQL es compatible con una amplia variedad de sistemas operativos, incluyendo Linux, Windows y MacOS. También es compatible con muchos lenguajes de programación, incluyendo PHP, Java, Python, Ruby y muchos más.
- **Facilidad de uso**: MySQL es relativamente fácil de usar. Tiene una sintaxis simple y un potente motor SQL para manipular datos.

Para los frameworks de front y back-end utilizaremos React para front y Node.js para back, las razones esenciales que tenemos para utilizar estos frameworks se basan en que son las herramientas más recomendadas y más utilizadas en base a el lenguaje de programación utilizado, cada uno de estos frameworks complementa y amplia las mecánicas que se pueden utilizar en ambos lados del programa. Por ultimo necesitamos utilizar un IDE para poder desarrollar el programa en toda su modalidad, el IDE que se utilizara es Visual Studio Code, esto por que es el IDE más amigable y sencillo para poder utilizar, además de que este IDE posee una amplia gama de compatibilidad con los lenguajes de programación en la actualidad, utilizando los plugines que también posee se pueden tener muchas ventajas a la hora de codificar aplicativos.