Desarrollo de Proyecto ADA 3ra entrega

Escrito Por Nexus

Especificación de Requisitos según el estándar de IEEE 830

Versión: 1.3

Fecha:25/10/24

Índice

1- **Caja Blanca (o Caja de Cristal)** – pagina 3

2- **Caja Negra** – pagina 4

3- **Valores Límites con Juegos de Datos** – pagina 5

4- **Plan de Testing** – pagina 6

5- **Justificación Caja Blanca** – pagina 8

6- **Casos de Prueba con Juegos de Datos** – pagina 9

7- **Manuales de Manipulación por Perfiles de Usuario** – pagina 10

8- **Manual de Instalación del Sistema** – pagina 11

9- **Manuales de Administración del Sistema** – pagina 14

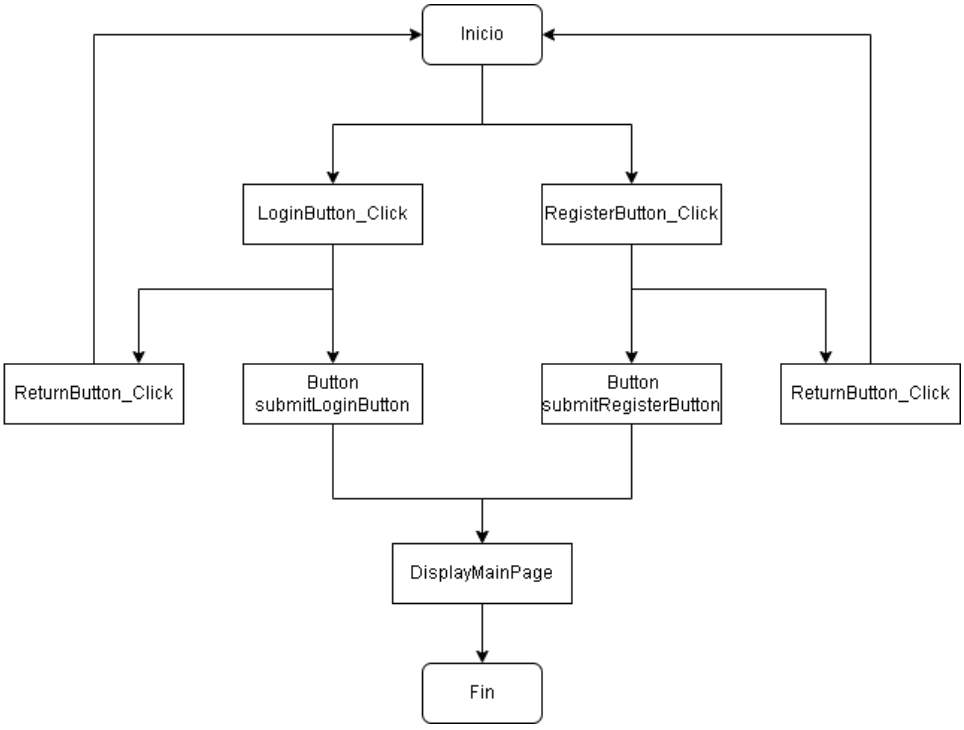
10- **Referencias** – pagina 18

11- **Enlaces** – pagina 19

1. Caja Blanca (o Caja de Cristal)

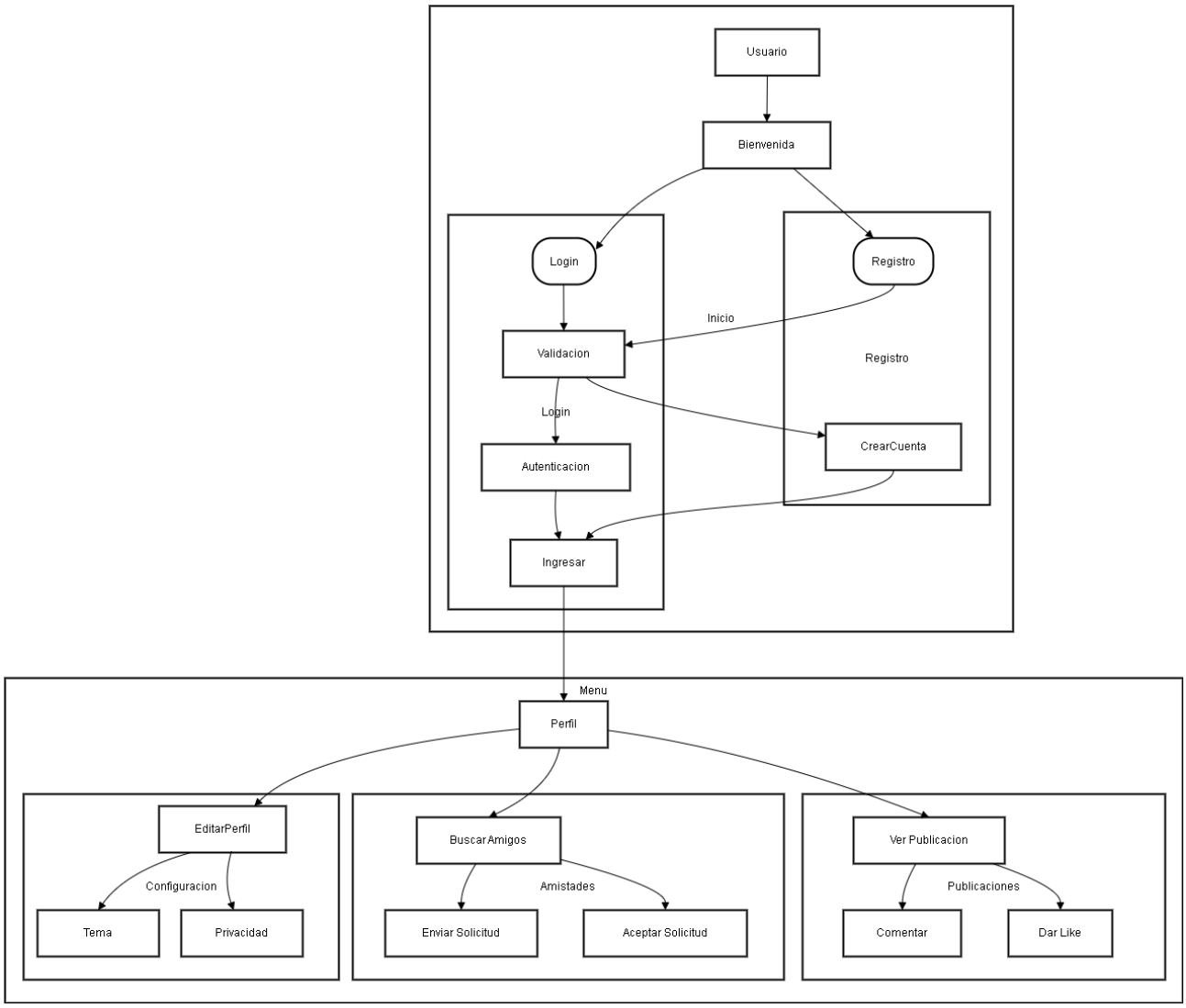
- **Descripción:** La prueba de Caja Blanca (o Caja de Cristal) permite ver y evaluar el código de la aplicación directamente para comprobar el flujo de datos y lógica interna.
- **Áreas de enfoque en Nexa:**
    - **Autenticación de Usuarios:** Verificar el flujo de autenticación (condiciones de login y logout) y el control de accesos para cada perfil de usuario.
    - **Manejo de Permisos:** Validar que los usuarios (Moderadores, admins de Comunidad, y Admins de App) tienen los permisos adecuados y no pueden acceder a funciones fuera de su rol.
    - **Navegación de Menú:** Comprobar las rutas entre los menús principales, comunidades y chats, confirmando que cada pantalla lleva a la vista esperada.
- **Justificación de Caja Blanca:** Se aplica para validar la lógica de permisos y flujos de acceso que controlan las acciones de cada perfil y sus módulos. Este método es esencial para asegurar que los usuarios no sobrepasen los permisos establecidos, y que los flujos de datos y control respondan a las condiciones definidas en el código.
- **Ejemplo de Caso de Prueba en Caja Blanca:**

LoginButton_Click
- Carga correctamente el apartado de logueo
RegisterButton_Click
- Carga correctamente el apartado de registro
Button submitLoginButton
- Carga correctamente el main page
Button submitRegisterButton
- Carga correctamente el main page
ReturnButton_Click
- Regresa correctamente al inicio - El logo desaparece al regresar
DisplayMainPage
- Carga correctamente



2. Caja Negra

- **Descripción:** Las pruebas de Caja Negra evalúan las funcionalidades externas de Nexa, enfocándose en entradas y salidas sin revisar el código fuente.
- **Áreas de enfoque en Nexa:**
    - **Login y Registro:** Prueba de inputs válidos e inválidos para el acceso.
    - **Chats Privados:** Verificar envío y recepción de mensajes en chats privados entre usuarios.
    - **Visualización de Posts en Comunidades:** Asegurar que los posts de las comunidades se muestran correctamente y en tiempo real.
- **Caja Negra de Interfaz:** Aquí se verificarán aspectos de usabilidad y navegación en la interfaz:
    - **Flujo de Menú:** Que el usuario pueda acceder intuitivamente a todas las secciones (amigos, comunidades, y solicitudes).
    - **Respuestas de Botones:** Comprobar que todos los botones de interfaz responden adecuadamente, excepto el de llamada, que actualmente es decorativo.
- **Ejemplo de Caso de Prueba en Caja Negra:**



3. Valores Límites con Juegos de Datos

- **Descripción:** Los valores límites identifican los extremos de entradas válidas para asegurarse de que la aplicación maneje adecuadamente los casos extremos.
- **Ejemplo en Nexa:**
    - **Longitud de Usuario y Contraseña:** Validar que el sistema acepte y rechace valores en los límites (mínimo y máximo de caracteres).
    - **Número de Amigos y Comunidades:** Asegurar que el sistema gestione correctamente listas con el número máximo de amigos o comunidades.
- **Ejemplo de Pruebas de Valores Límites:**

Campo	Limite Inferior Invalido	Limite Inferior Valido	Limite Superior Valido	Limite Superior Invalido
Nombre de Usuario	4 Caracteres (falla)	5 Caracteres (pasa)	20 Caracteres (pasa)	21 Caracteres (falla)
Contraseña	4 Caracteres (falla)	5 Caracteres (pasa)	20 Caracteres (pasa)	21 Caracteres (pasa)
Numero de Amigos	-	0 Amigos (pasa)	500 Amigos (pasa)	501 Amigos (pasa)
Mensaje Privado	0 Caracteres (opcional)	1 Carácter (pasa)	500 Caracteres (pasa)	501 Caracteres (falla)
Comunidades Unidas	-	0 Comunidades (pasa)	100 Comunidades (pasa)	101 Comunidades (falla)
Edad del Usuario	12 años (falla)	13 años (pasa)	100 años (pasa)	101 años (falla)

4. Plan de Testing

Objetivo

El objetivo del testing es verificar que todas las funcionalidades principales de Nexa se comporten según lo esperado, asegurando que el sistema sea robusto, seguro y eficiente. Esto incluye validar el **login**, el **registro de usuarios**, la gestión de **amigos**, el funcionamiento de las **comunidades**, y la correcta **administración de permisos**. La finalidad es que los usuarios puedan interactuar con el sistema sin problemas, mientras que los administradores puedan gestionar la plataforma sin inconvenientes.

Alcance

El alcance de este plan de testing abarca las funciones clave de la aplicación, específicamente:

- **Módulo de Login:** Verificar que el proceso de autenticación sea seguro y fluido para los usuarios.
- **Registro de Usuario:** Asegurar que los nuevos usuarios puedan registrarse correctamente, validando datos y asegurando la creación adecuada de cuentas.
- **Gestión de Amigos:** Comprobar que los usuarios puedan agregar, eliminar y gestionar correctamente a sus amigos en la plataforma.
- **Comunidades:** Validar la creación de comunidades, la gestión de miembros y las interacciones dentro de estas.
- **Administración de Permisos:** Asegurar que los roles y permisos de usuario sean gestionados de manera correcta y segura, permitiendo que los administradores controlen el acceso a diversas funcionalidades.

Tipos de Prueba

El proceso de testing se llevará a cabo utilizando diferentes enfoques para cubrir todos los aspectos del sistema, incluyendo pruebas de seguridad, funcionalidad y rendimiento. Los tipos de pruebas que se realizarán son los siguientes:

- **Caja Blanca:** Se realizarán pruebas a nivel de código y lógica interna, asegurando que los procesos de back-end, como las interacciones con la base de datos, se ejecuten correctamente.
- **Caja Negra:** Se probarán las funcionalidades del sistema desde la perspectiva del usuario, sin conocer el código fuente, para asegurar que los flujos de trabajo sean intuitivos y funcionen sin errores.
- **Pruebas Unitarias:** Cada componente del sistema será probado individualmente para asegurar que cada función y módulo ejecute su tarea correctamente.
- **Pruebas de Integración:** Se validará la correcta interacción entre los diferentes módulos del sistema, como el proceso de login y la asignación de permisos, asegurando que todo el flujo de trabajo esté interconectado adecuadamente.
- **Pruebas de Aceptación:** Finalmente, se llevará a cabo una prueba para asegurar que el sistema cumpla con los criterios establecidos por los usuarios y stakeholders, y que esté listo para su implementación.

Criterios de Aceptación

Para que el sistema pase la fase de testing, se requiere que al menos el **90% de las funciones críticas** superen las pruebas sin fallos. Esto incluye funcionalidades como el **login**, la **gestión de permisos** y la **interacción en las comunidades**. Cualquier fallo crítico o bloqueo del sistema será reportado y corregido antes de que el sistema sea considerado apto para producción.

Cronograma

El plan de testing se llevará a cabo en **tres fases** a lo largo de dos semanas:

- **Fase 1 (Pruebas Unitarias):** Se ejecutarán pruebas unitarias en cada uno de los módulos del sistema del 1 al 4 de noviembre.
- **Fase 2 (Pruebas de Integración):** Durante esta fase, se validará la integración de los módulos y su comunicación entre sí, del 5 al 9 de noviembre.
- **Fase 3 (Pruebas de Sistema):** Finalmente, se realizarán pruebas de aceptación y de sistema en conjunto, verificando el funcionamiento integral de Nexa, del 10 al 14 de noviembre.

5. Justificación de Caja Blanca

La técnica de **Caja Blanca** se emplea para examinar y validar la lógica interna del sistema, permitiendo una revisión exhaustiva de los flujos de control y los procesos que gestionan los datos dentro de la plataforma. Su principal objetivo es garantizar que los algoritmos, estructuras de datos y la interacción entre los diferentes módulos de Nexa se ejecuten de manera eficiente y segura. Esta metodología es fundamental para identificar posibles vulnerabilidades en la seguridad, errores en la manipulación de datos o cualquier fallo en la lógica que pueda afectar la operativa del sistema.

A través de la Caja Blanca, es posible llevar a cabo un análisis detallado de los componentes internos de Nexa, como el proceso de **autenticación** de usuarios, el **control de acceso** mediante la asignación de permisos y la correcta **interacción entre módulos** del sistema. Estos aspectos son cruciales para asegurar que cada flujo de trabajo se ejecute correctamente y que no existan puntos débiles que puedan ser explotados por posibles amenazas externas.

Además, el uso de la Caja Blanca permite realizar pruebas en diferentes niveles de granularidad, desde el código fuente hasta los procesos más amplios, lo que facilita la detección de problemas a nivel de **lógica de negocio**, **seguridad** y **funcionalidad**. Esto resulta especialmente valioso para sistemas que manejan información sensible, como es el caso de Nexa, ya que ofrece una visión clara de cómo los datos se gestionan en su interior y permite anticipar posibles fallos antes de que afecten a los usuarios o a la operación del sistema.

En resumen, la aplicación de la Caja Blanca no solo es una medida preventiva frente a fallos técnicos o brechas de seguridad, sino que también contribuye a **optimizar el rendimiento** y **fiabilidad** del sistema, asegurando que Nexa funcione de manera robusta y eficiente en todos sus aspectos operativos. Su implementación es esencial para validar que todos los componentes del sistema interactúan correctamente, minimizando riesgos y garantizando un entorno seguro y funcional para los usuarios.

6. Casos de Prueba con Juegos de Datos

ID de Prueba	Campo	Descripcion	Entrada de Prueba	Resultado Esperado
1	Nombre de Usuario	Verificar que falla con menos de 3 caracteres	“AB”	Falla (Demasiado Corto)
2	Nombre de Usuario	Verificar que pasa con el limite superior de 20 caracteres	“UsuarioValidoLargo123”	Pasa (Nombre Valido)
3	Contraseña	Verificar que falla con menos de 8 caracteres	“1234567”	Falla (Demasiado Corto)
4	Contraseña	Verificar que pasa con el limite superior de 50 caracteres	“ContraseñaSeguraConLongitudMaximaValida12345”	Pasa (Contraseña Valida)
5	Numero de Amigos	Verificar que pasa con 0 amigos	0	Pasa (Sin Amigos)
6	Numero de Amigos	Verificar que falla con mas de 500 amigos	501	Falla (Excede el Limite Permitido)
7	Mensajes Privados	Verificar que pasa con un mensaje vacio	-	Pasa (Mensaje Vacio Opcional)
8	Mensajes Privados	Verificar que falla con mas de 500 caracteres	“a” * 501	Falla (Mensaje Demasiado Largo)
9	Comunidades Unidas	Verificar que pasa con 0 comunidades	0	Pasa (Sin Comunidades)
10	Comunidades Unidas	Verificar que falla con mas de 100 comunidades	101	Falla (Excede el Limite de Comunidades)

7. Manuales de Manipulación por Perfiles de Usuario

- **Usuario Normal:**
  El manual para el **Usuario Normal** está orientado a aquellas personas que utilizan la plataforma de manera regular, sin privilegios administrativos. Este manual incluye instrucciones detalladas sobre cómo **registrarse** en el sistema, **configurar su perfil** personal, y cómo navegar en la plataforma de manera efectiva. Además, cubre aspectos clave como **agregar amigos**, **unirse a comunidades** y **participar en discusiones**. También incluye guías sobre cómo **enviar mensajes privados** a otros usuarios, responder a comentarios, y cómo gestionar sus notificaciones para mantenerse al tanto de las actividades relevantes dentro de las comunidades a las que pertenece. Todo esto está explicado de manera sencilla, para garantizar que los usuarios nuevos puedan familiarizarse rápidamente con las funciones básicas de la plataforma sin complicaciones.
- **Admin de Comunidad:**
  El **manual para Admin de Comunidad** está dirigido a aquellos usuarios con responsabilidades administrativas dentro de las comunidades específicas en la plataforma. Este manual detalla cómo **crear nuevas comunidades**, establecer sus normas y personalizar los ajustes de cada una, como el tipo de contenido permitido o las configuraciones de privacidad. También explica cómo **asignar moderadores** y **gestionar roles** dentro de la comunidad para garantizar un control adecuado sobre el contenido y las interacciones entre los miembros. Además, cubre cómo gestionar y **moderar publicaciones**, aprobando o rechazando contenido según las reglas de la comunidad, así como la capacidad de **eliminar comentarios o bloquear usuarios** que infringen las normas. Este manual es crucial para asegurar que los administradores de comunidad puedan mantener un ambiente organizado, seguro y alineado con los valores de la plataforma.
- **Moderador:**
  El manual para **Moderadores** proporciona instrucciones claras sobre cómo llevar a cabo sus funciones dentro de las comunidades. Los moderadores tienen la responsabilidad de mantener el **orden y la calidad del contenido** que se publica en la plataforma. Este manual explica cómo **revisar publicaciones** de los usuarios, **aprobar o eliminar posts** que no cumplan con las normas establecidas, y cómo intervenir en conversaciones que puedan ser inapropiadas o que inciten a comportamientos perjudiciales. Además, incluye directrices sobre cómo gestionar **informes de usuarios** que puedan estar violando las normas y cómo actuar ante situaciones de conflicto entre miembros de la comunidad. El manual también aborda las mejores prácticas para mantener una comunicación efectiva con los usuarios, aclarar dudas y resolver disputas de manera justa y equilibrada, asegurando que la comunidad se mantenga sana y respetuosa.
- **Admin de App:**
  El **manual para Admin de App** está dirigido a los administradores del sistema que tienen la responsabilidad de gestionar y mantener la plataforma en su conjunto. Este manual es más técnico y detallado, y cubre todas las funcionalidades avanzadas necesarias para administrar la infraestructura y los usuarios de la plataforma. Aquí se incluyen instrucciones sobre cómo gestionar y **configurar la base de datos**, supervisar los **permisos de acceso** a diferentes partes del sistema, y **mantener el sistema operativo** para asegurar su rendimiento óptimo. El manual también explica cómo gestionar la **creación, modificación y eliminación de cuentas de usuario**, asegurando que cada uno tenga el acceso adecuado según su perfil y rol. Además, cubre la implementación de **actualizaciones de seguridad**, la **supervisión de logs de actividad** y cómo ejecutar tareas de **respaldo y recuperación de datos**. Este manual está orientado a asegurar que los administradores de la aplicación puedan mantener el sistema seguro, eficiente y libre de errores, y puedan gestionar los usuarios de manera efectiva.

8. Manual de Instalación del Sistema

Requisitos Previos

Antes de proceder con la instalación del sistema, es fundamental asegurarse de que se cumplen los siguientes requisitos previos. Estos componentes son esenciales para garantizar que el sistema se ejecute correctamente en el servidor.

1. **Visual Studio 2019**
   El sistema está desarrollado utilizando tecnologías compatibles con **Visual Studio 2019**, por lo que se requiere tener instalada esta versión del IDE. Visual Studio proporcionará las herramientas necesarias para la construcción, depuración y ejecución de la aplicación. Asegúrese de tener instalados los componentes de C# y .NET Core.
2. **MySQL**
   La base de datos que utiliza la aplicación es **MySQL**, por lo que será necesario contar con una instalación funcional de MySQL en el servidor. Además, se debe contar con las credenciales de acceso adecuadas para crear y gestionar las bases de datos relacionadas con el sistema.
3. **Extensión JSON en MySQL**
   La aplicación hace uso intensivo de estructuras de datos en formato JSON. Por lo tanto, es necesario tener habilitada la **extensión JSON** en MySQL para el manejo adecuado de este tipo de datos. Asegúrese de que su instalación de MySQL soporte esta extensión.
4. **Docker en Servidor Ubuntu**
   El sistema requiere el uso de **Docker** para la contenedorización de la aplicación. Docker permite ejecutar la aplicación de manera aislada y más eficiente en el servidor. Asegúrese de que Docker esté instalado en el servidor Ubuntu y que esté funcionando correctamente antes de proceder con la instalación de la aplicación.

Pasos de Instalación

A continuación, se detallan los pasos específicos que deben seguirse para instalar y configurar el sistema correctamente en el servidor.

1. Descargar los Archivos del Repositorio

El primer paso consiste en obtener todos los archivos necesarios para la instalación de la aplicación. Para ello, siga estos pasos:

- Dirígete al repositorio del proyecto en **GitHub** o el sistema de gestión de código fuente que se utilice.
- Clona el repositorio en el servidor utilizando Git. Abre una terminal y ejecuta el siguiente comando:

  ```
  git clone https://github.com/usuario/repositorio.git
  ```

- Una vez descargado el repositorio, navega hasta el directorio del proyecto utilizando el comando `cd` en la terminal.

2. Configurar la Base de Datos MySQL e Importar el Esquema

La aplicación requiere una base de datos MySQL para almacenar información persistente. Los pasos para configurar la base de datos son los siguientes:

- **Instalar MySQL** (si no está instalado):
  Si MySQL no está instalado en el servidor, puedes instalarlo ejecutando los siguientes comandos:

  ```
  sudo apt update
  sudo apt install mysql-server
  sudo mysql_secure_installation
  ```

- **Configurar la Base de Datos:**
  Una vez que MySQL esté instalado, debes crear una base de datos para el sistema. Puedes hacerlo accediendo a la terminal de MySQL y ejecutando:

  ```
  mysql -u root -p
  CREATE DATABASE sistema;
  ```

- **Importar el Esquema de la Base de Datos:**
  En el repositorio descargado, encontrarás un archivo SQL que contiene el esquema de la base de datos necesario para la aplicación. Importa el esquema utilizando el siguiente comando:

  ```
  mysql -u root -p sistema < /ruta/del/repositorio/esquema.sql
  ```

  Esto creará las tablas y estructuras necesarias en la base de datos. Asegúrate de que la configuración de la base de datos en el archivo de configuración de la aplicación apunte a la base de datos correcta.

3. Configurar Docker en el Servidor Ubuntu y Cargar la Aplicación

La aplicación se ejecutará dentro de un contenedor Docker, lo que simplifica la gestión del entorno y asegura la consistencia entre diferentes servidores. Para configurar Docker y cargar la aplicación, sigue estos pasos:

- **Instalar Docker en Ubuntu:**
  Si Docker no está instalado, puedes instalarlo con los siguientes comandos:

  ```
  sudo apt update
  sudo apt install apt-transport-https ca-certificates curl software-
  properties-common
  curl -fsSL https://get.docker.com -o get-docker.sh
  sudo sh get-docker.sh
  ```

- **Verificar la Instalación de Docker:**
  Para asegurarte de que Docker se instaló correctamente, ejecuta:

  ```
  docker --version
  ```

- **Construir y Cargar la Aplicación en Docker:**
  Una vez que Docker esté instalado, navega hasta el directorio del repositorio donde se encuentran los archivos Docker y ejecuta los siguientes comandos para construir la imagen del contenedor y luego ejecutarlo:

  ```
  sudo docker build -t nombre-imagen .
  sudo docker run -d -p 80:80 nombre-imagen
  ```

Esto construirá el contenedor con la aplicación y lo pondrá en ejecución, asignando el puerto 80 del contenedor al puerto 80 de tu servidor.

4. Ejecutar la Aplicación en Visual Studio para Verificar la Conexión a la Base de Datos

Una vez que el contenedor Docker esté en funcionamiento, el siguiente paso es verificar que la aplicación esté correctamente conectada a la base de datos y funcionando sin errores. Para ello:

- Abre **Visual Studio 2019** en tu máquina local.
- **Clona nuevamente el repositorio** en tu entorno de desarrollo local (si aún no lo has hecho).
- Abre la solución en Visual Studio y **configura las cadenas de conexión** a la base de datos en el archivo de configuración de la aplicación (por lo general, en `appsettings.json`).
- Asegúrate de que las credenciales y la IP del servidor MySQL estén configuradas correctamente.
- **Ejecuta la aplicación** en modo depuración para verificar que la conexión a la base de datos se establezca correctamente y que la aplicación se inicie sin problemas.

Verificación Final

Una vez que hayas completado todos los pasos de instalación, asegúrate de que:

- La aplicación se ejecuta correctamente en Docker y es accesible desde un navegador web.
- La base de datos está en funcionamiento y la aplicación puede interactuar con ella sin problemas.
- Las configuraciones de seguridad y permisos de acceso están correctamente aplicadas.

Si la aplicación se ejecuta correctamente y la base de datos se conecta sin problemas, el proceso de instalación habrá sido exitoso.

9. Manuales de Administración del Sistema

Funciones de Administración

Las funciones de administración del sistema son esenciales para la gestión y el control de los aspectos clave de la plataforma, como la creación de comunidades, la asignación de permisos de usuario y el mantenimiento de la base de datos. A continuación, se detallan las principales tareas de administración que deben realizarse para mantener el sistema operativo de manera eficiente:

1. Creación y Eliminación de Comunidades

Uno de los aspectos fundamentales de la administración del sistema es la gestión de **comunidades**. Los administradores tienen la capacidad de crear nuevas comunidades, establecer sus reglas, gestionar sus miembros y moderar el contenido. Los pasos para crear una comunidad son los siguientes:

- **Creación de una Comunidad:**
  - Ingrese a la **sección de administración** en el panel de control del sistema.
  - Seleccione la opción de **"Crear nueva comunidad"**.
  - Complete los campos requeridos, como el nombre de la comunidad, la descripción, las categorías o temas de interés y las configuraciones de privacidad (comunidad abierta o cerrada).
  - Determine si la comunidad será pública o privada, y configure las opciones de acceso (por invitación, membresía abierta, etc.).
  - Establezca las normas básicas de la comunidad para asegurar un entorno respetuoso y organizado.
- **Eliminación de una Comunidad:**
  - Acceda al panel de administración y seleccione la comunidad que desea eliminar.
  - Verifique que tiene los permisos necesarios para eliminar la comunidad.
  - Elimine la comunidad de manera definitiva, lo que también puede implicar la eliminación de los datos asociados, como publicaciones y usuarios vinculados.

2. Ajuste de Permisos de Usuario

La gestión de permisos es clave para garantizar que los usuarios solo tengan acceso a las funcionalidades que les correspondan según su rol (usuario normal, moderador, administrador, etc.). Los pasos para ajustar los permisos son los siguientes:

- **Asignación de Roles y Permisos:**
  - Acceda al panel de administración de usuarios.
  - Seleccione el usuario al que desea asignar un rol específico (por ejemplo, moderador, miembro, o administrador).
  - Defina los permisos de ese rol, tales como la capacidad de crear publicaciones, moderar comentarios, acceder a funciones administrativas, etc.
  - Asegúrese de que los permisos sean apropiados para evitar que los usuarios accedan a funciones sensibles para las que no tienen autorización.
- **Modificación de Permisos:**
  - Si es necesario modificar los permisos de un usuario, acceda nuevamente al panel de administración, seleccione al usuario y ajuste sus permisos según los cambios que desee implementar.

    - Si un usuario cambia de rol, asegúrese de que las nuevas configuraciones de permisos estén alineadas con sus nuevas responsabilidades.
- **Revocación de Permisos:**
    - Si un usuario ya no necesita acceso a ciertas funcionalidades, puede revocar sus permisos a través del panel de administración. Esto es especialmente importante en casos de abuso o cuando un usuario ya no forma parte de la plataforma.

3. Mantenimiento de Bases de Datos

El mantenimiento de la base de datos es un aspecto fundamental para garantizar la integridad y el rendimiento del sistema. Las tareas clave de mantenimiento incluyen la optimización de consultas, la actualización de esquemas y la gestión de respaldos.

- **Optimización de la Base de Datos:**
    - Revise periódicamente las **consultas SQL** para asegurarse de que están optimizadas. El uso de índices adecuados y la normalización de la base de datos pueden mejorar significativamente el rendimiento de las consultas.
    - Verifique las tablas que tienen un gran volumen de datos y considere la posibilidad de realizar particiones o utilizar técnicas de optimización como la desnormalización si es necesario.
- **Actualización del Esquema de la Base de Datos:**
    - Las actualizaciones de la base de datos pueden ser necesarias a medida que el sistema evoluciona. Asegúrese de contar con un **plan de migración de datos** para realizar cambios en el esquema sin afectar a los usuarios.
    - Utilice herramientas de migración de bases de datos como **Flyway** o **Liquibase** para automatizar las actualizaciones del esquema de la base de datos y garantizar que las migraciones se realicen de manera coherente.
- **Monitoreo y Administración de la Base de Datos:**
    - Realice un monitoreo continuo de la base de datos para detectar problemas de rendimiento, cuellos de botella o posibles caídas del sistema.
    - Revise los **logs** de la base de datos para detectar errores y problemas de seguridad.

Respaldos y Recuperación de Datos

Los respaldos regulares y la capacidad de recuperación son esenciales para garantizar la disponibilidad y seguridad de los datos en caso de fallos o pérdidas. A continuación, se describen los procedimientos para realizar respaldos y la recuperación de datos:

1. Proceso de Respaldo en Docker

En el caso de que el sistema se ejecute dentro de contenedores Docker, el respaldo de los datos y el contenedor completo es fundamental para restaurar la aplicación en caso de una caída del sistema. El proceso de respaldo implica dos pasos principales: respaldar los datos y respaldar la imagen del contenedor.

- **Respaldo de Datos:**
    - Utilice el comando `docker exec` para acceder al contenedor que contiene la base de datos y realice un **volcado** de la base de datos MySQL utilizando `mysqldump`:

      ```
      docker exec -i nombre_del_contenedor mysqldump -u root -p nombre_base_de_datos > respaldo.sql
      ```

    - Este comando crea una copia de seguridad completa de la base de datos en un archivo `.sql`.
- **Respaldo del Contenedor Docker:**
    - Si desea hacer un respaldo completo del contenedor y su configuración, puede crear una **imagen Docker** del contenedor actual:

      ```
      docker commit nombre_del_contenedor nombre_imagen_respaldo
      docker save -o respaldo.tar nombre_imagen_respaldo
      ```

    - Esto crea un archivo comprimido con la imagen del contenedor, que puede ser almacenado y utilizado para restaurar el contenedor en otro servidor si es necesario.

2. Recuperación de Datos en Caso de Pérdida

En caso de que se pierdan los datos debido a un fallo del sistema o una corrupción de la base de datos, es crucial tener un procedimiento claro de recuperación:

- **Restauración de la Base de Datos:**
    - Para recuperar la base de datos desde un archivo de respaldo `.sql`, ejecute el siguiente comando en el servidor MySQL:

      ```
      mysql -u root -p nombre_base_de_datos < respaldo.sql
      ```

- **Restauración del Contenedor Docker:**
    - Para restaurar el contenedor desde una imagen de respaldo, primero cargue la imagen en Docker:

      ```
      docker load -i respaldo.tar
      ```

    - Luego, ejecute un nuevo contenedor a partir de la imagen restaurada:

      ```
      docker run -d -p 80:80 nombre_imagen_respaldo
      ```

Optimización del Sistema

La optimización del sistema es crucial para garantizar un rendimiento adecuado, especialmente en entornos con recursos limitados o cuando se enfrenta a una alta demanda de usuarios. A continuación, se ofrecen algunas recomendaciones clave para optimizar el sistema:

1. Optimización en Máquinas de Bajos Recursos

Si el sistema se va a ejecutar en un servidor con **recursos limitados** (memoria RAM, CPU, etc.), es importante tomar medidas para reducir el consumo de recursos:

- **Uso de Docker para Aislamiento de Recursos:**
    - Configure Docker para limitar el uso de CPU y memoria mediante las opciones `--memory` y `--cpus`. Esto permite que el contenedor no consuma más recursos de los que el servidor puede manejar.
- **Optimización de Consultas SQL:**
    - Evite consultas complejas o que requieran de grandes volúmenes de datos para procesarse. Las consultas optimizadas, el uso de índices y la normalización de la base de

datos pueden mejorar significativamente el rendimiento en sistemas con recursos limitados.

- **Uso de Caché:**
  - Implementar **caché** para reducir la carga en la base de datos y mejorar la velocidad de acceso a los datos más frecuentemente consultados.

2. Pruebas en Distintos Entornos de Usuario

Realizar **pruebas de carga** y **pruebas de rendimiento** es clave para identificar los cuellos de botella del sistema antes de que se presenten problemas en entornos de producción.

- **Pruebas de Carga:**
  - Realice pruebas de carga en entornos controlados para simular diferentes niveles de tráfico y comportamiento de los usuarios. Herramientas como **Apache JMeter** o **Gatling** pueden ser útiles para generar tráfico simulado y evaluar el rendimiento.
- **Pruebas de Compatibilidad:**
  - Asegúrese de que el sistema sea compatible con distintos entornos, desde dispositivos móviles hasta navegadores web, y que se comporte de manera eficiente incluso en configuraciones de hardware y red más limitadas.

10. Referencias

1. **Sommerville, Ian** - *Ingeniería de Software*, 10ª Edición. Un clásico en la disciplina de ingeniería de software que cubre metodologías de testing, diseño y gestión de proyectos.
2. **Pressman, Roger** - *Ingeniería de Software: Un Enfoque Práctico*, 7ª Edición. Incluye explicaciones detalladas de las pruebas de software, incluyendo Caja Blanca y Caja Negra.
3. **Beizer, Boris** - *Software Testing Techniques*, 2ª Edición. Este libro aborda métodos de prueba de software a profundidad, proporcionando ejemplos claros y estrategias para pruebas de Caja Blanca y Caja Negra.
4. **MySQL Documentation** - *Manual de Usuario MySQL*, disponible en línea. Contiene referencias y ejemplos para la configuración y administración de bases de datos, ideal para el manual de instalación.
5. **Patton, Ron** - *Software Testing*, 2ª Edición. Introducción accesible a los principios de pruebas de software, incluyendo la planificación y diseño de casos de prueba.
6. **Cem Kaner** - *Testing Computer Software*. Este libro profundiza en la teoría y la práctica de testing, con un enfoque en pruebas de Caja Negra y límites de datos.
7. **Knott, Adam** - *The Complete Guide to Test Automation*. Guía detallada sobre la automatización de pruebas en aplicaciones.

11. Enlaces

1. **Testing en C#:**
    - Microsoft Docs - Testing en .NET - Documentación oficial de Microsoft sobre cómo realizar pruebas en aplicaciones .NET.
2. **MySQL y Docker:**
    - Docker Documentation - Getting Started - Guía oficial de Docker que ayuda a configurar contenedores en Ubuntu.
    - MySQL Docker Container Setup - Repositorio de Docker Hub para configurar un contenedor de MySQL.
3. **Casos de Prueba de Caja Blanca y Caja Negra:**
    - Ministry of Testing - Black Box and White Box Testing - Portal con artículos y guías prácticas sobre técnicas de prueba.
    - Guru99 - White Box Testing - Ejemplos de testing de Caja Blanca y métodos de validación de código.
    - Guru99 - Black Box Testing - Explicación detallada de técnicas de prueba de Caja Negra.
4. **Manejo de Límites y Juegos de Datos:**
    - Software Testing Fundamentals - Boundary Value Analysis - Explicación de cómo aplicar análisis de valores límites con ejemplos.
5. **Manual de Usuario y Administración:**
    - GitBook - Herramienta en línea para crear manuales detallados de usuarios y administración, permitiendo compartir guías de instalación y administración.
    - Confluence by Atlassian - Plataforma colaborativa para documentar proyectos y guías de uso.
6. **Pruebas de Interfaz de Usuario (UI Testing):**
    - Selenium for Windows Forms - Herramienta para automatizar pruebas de interfaces gráficas, aunque originalmente para web, tiene extensiones que soportan aplicaciones de Windows Forms.
    - Appium - Plataforma de pruebas para aplicaciones de escritorio y móviles, útil ya que Nexa tiene componentes en Windows Forms.
7. **Mejores Prácticas para el Desarrollo en Visual Studio:**
    - Microsoft Visual Studio - Guía de Mejores Prácticas – Lugar donde se desarrollo Nexa en Visual Studio.
8. **Gestión y Optimización de Docker en Ubuntu Server:**
    - Digital Ocean - Docker en Ubuntu Server - Instrucciones específicas para instalar y configurar Docker en Ubuntu Server.