PROYECTO FINAL PYTHON DJANGO:

PROFESOR: HANZ SAENZ

PROYECTO NAMISTASK

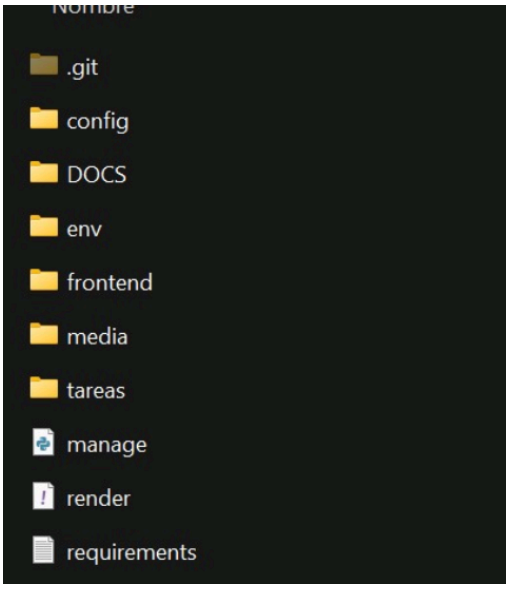
INTEGRANTES PROYECTO :

Brenda Nathalia Domínguez Rojas

Michael Fernando Rodríguez Melo

Juan Sebastian Ardila Rico

Observación : Los documentos requeridos para este proyecto se encuentran dentro de la carpeta DOCS,la cual está dentro de github adicional el enlace del proyecto se encuentra dentro de esta carpeta , dado que usó el mismo metodo del proyecto anterior



Enlace Github: https://github.com/Maicfer/namitask.git

GUIA PARA DESPLIEGUE
PROYECTO

NAMISTASK

OBJETIVO

- Definir Procedimientos Claros: Establecer un conjunto de pasos y directrices precisos para el despliegue de la aplicación *Namitask* en un entorno remoto, desde la preparación del servidor hasta la verificación post-despliegue.
- Garantizar la Eficiencia y Seguridad: Asegurar que el procedimiento de instalación, configuración y lanzamiento sea reproducible, minimizando tiempos de inactividad y garantizando un entorno seguro.
- Facilitar la Administración Operativa: Proveer a los equipos de desarrollo, administración de sistemas y DevOps una referencia actualizable y confiable para gestionar la infraestructura en producción.
- Optimizar el Procesos de Actualización: Permitir la integración contínua de mejoras y correcciones, estableciendo un marco para procedimientos de rollback en caso de fallas.

NECESIDAD

- Complejidad del Despliegue Remoto: La administración de entornos remotos implica múltiples capas de configuraciones y dependencias que requieren documentación detallada para evitar errores y garantizar la estabilidad del servicio.
- Garantía de Continuidad del Servicio: Es imprescindible contar con un manual que permita implementar cambios y actualizaciones sin interrumpir la operatividad de la aplicación, asegurando que cada paso se realice de forma controlada.

1.1 MANUAL DE USO

Namitask es una aplicación moderna de gestión de tareas diseñada para optimizar la organización de actividades y fomentar la colaboración en equipo. Inspirada en metodologías ágiles, la herramienta te permite visualizar tus tareas mediante un intuitivo tablero Kanban, acceder a detalles de cada tarea, y gestionar tu perfil de usuario de forma sencilla.

2. Requisitos y Acceso

- **Requisitos del Navegador y Sistema:**
  Asegúrate de usar un navegador actualizado para una experiencia óptima (Chrome, Firefox, Edge, etc.).
- **Acceso a la Aplicación:**
  Ingresa a la url oficial de Namitask e inicia sesión o regístrate para crear tu cuenta.
- **Credenciales Iniciales (para pruebas):**
  Si se trata de una demo o versión beta, se te proporcionarán credenciales temporales.

3. Inicio de Sesión y Registro

Esta sección cubre la autenticación y creación de usuario:

- **Registro (Registro.jsx):**
    - Rellena el formulario de registro con tus datos personales.
    - Confirma tu correo (si aplica) para validar la cuenta.
- **Inicio de Sesión (Login.jsx):**
- Ingresa tus credenciales en el formulario de login.
- Ante un error, verás mensajes de aviso para corregir tus datos.

Nota: El proceso de autenticación garantiza el acceso seguro a las rutas privadas (implementadas en RutaPrivada.jsx), protegiendo la información de cada usuario.

4. Navegación en la Aplicación

La aplicación cuenta con una **barra de navegación (Navbar.jsx)** que te facilita acceder a las diferentes secciones:

- **Dashboard:**
  Aquí se agrupan tus tareas y se presenta un resumen general de tu actividad.
- **Perfil (Profile.jsx):**
  Administra tu información personal y preferencias de la cuenta.
- **Acceso a Funcionalidades:**
  La navegación dinámica te dirige a diferentes módulos, asegurando que solo los usuarios autenticados puedan ver contenidos sensibles.

5. Gestión de Tareas

Namitask se centra en la administración efectiva de tus tareas. Esta sección explica cómo realizar las acciones principales:

- **Visualización del Tablero Kanban (TableroKanban.jsx):**
    - Visualiza las tareas organizadas en columnas (por ejemplo, "Por hacer", "En proceso" y "Completadas").
    - Arrastra y suelta tareas entre columnas para actualizar su estado.
- **Detalles y Edición de Tareas:**
- **Visualización de Detalles (TareaDetalle.jsx):** Consulta información completa de cada tarea.
- **Edición de Tareas (EditarTarea.jsx):** Accede a esta funcionalidad para modificar datos, cambiar prioridades o actualizar estados.
- **Gestión de Archivos Adjuntos (TareaAdjunta.jsx):** Añade o consulta archivos relacionados con cada tarea.
- **Etiquetas y Categorías (Etiquetas.jsx):** Organiza tareas mediante etiquetas que facilitan la búsqueda y la clasificación.

6. Funcionalidades Avanzadas

Para usuarios que desean aprovechar al máximo las capacidades de Namitask, se incluyen funcionalidades pensadas para mejorar la experiencia:

- **Rutas Privadas (RutaPrivada.jsx):**
  Estas rutas garantizan que solo los usuarios autenticados puedan acceder a funciones críticas o datos sensibles.
- **Personalización y Ajustes Avanzados:**
  Accede a configuraciones disponibles en el sistema (dentro de la estructura de *services* y *context*) que permiten ajustar la aplicación según tus necesidades.

7. Solución de Problemas y Preguntas Frecuentes (FAQ)

- **Errores Comunes:**
  - Problemas al iniciar sesión: revisa que tus credenciales sean correctas.
  - Fallas en la carga del tablero: comprueba tu conexión a Internet y actualiza la página.
- **Preguntas Frecuentes:**
  Incluye respuestas a las dudas más comunes (por ejemplo, “¿Cómo se crea una nueva tarea?”, “¿Cómo cambio mi contraseña?”).
- **Soporte Técnico:**
  Si el problema persiste, contacta al equipo de soporte a través del correo o canal designado.

8. Actualizaciones y Mantenimiento

- **Actualizaciones:**
  Namitask se actualiza periódicamente. Recibirás notificaciones o verás mensajes en la aplicación cuando una nueva versión esté disponible.
- **Recomendaciones de Mantenimiento:**
  Asegúrate de tener siempre tu aplicación con la versión más reciente para disfrutar de mejoras y parches de seguridad.

9. Glosario y Apéndices

- **Glosario de Términos:**
  - **Kanban:** Método para la gestión visual de tareas.
  - **Ruta Privada:** Función de seguridad que limita el acceso a usuarios autenticados.
- **Apéndices:**
  Se pueden incluir diagramas adicionales, ejemplos de uso avanzado o documentación técnica para usuarios interesados en integraciones adicionales.

Referencia Técnica: Organización del Proyecto

El manual se apoya en la arquitectura interna de Namitask para facilitar su mantenimiento y evolución:

- **Carpeta `frontend/src/components`:**
  Aquí se encuentran los distintos componentes de la interfaz, cada uno con una responsabilidad específica (por ejemplo, edición de tareas, gestión de la barra de navegación, etc.).

- **Carpeta `frontend/src/pages`:**
  Concentra las vistas principales, como el Dashboard.
- **Carpeta `frontend/src/services`:**
  Alberga archivos de estilos, configuración general y puntos de entrada de la aplicación.

Esta organización modular permite que cada funcionalidad esté claramente definida, lo que a su vez facilita la actualización tanto del manual de uso como del software en sí.

1. **2Manual de Despliegue Remoto – Namitask**

1. Introducción

Este manual tiene como objetivo describir, paso a paso, el proceso para desplegar remotamente la aplicación *Namitask*. Se abordan desde la preparación del entorno en el servidor hasta la verificación de un despliegue exitoso, pasando por la instalación de dependencias, configuración del entorno y puesta en marcha de servicios críticos. Está dirigido a desarrolladores y administradores de sistemas que administran la infraestructura de la aplicación.

2. Requisitos Previos

Antes de iniciar el despliegue, asegúrate de contar con lo siguiente:

- **Servidor Remoto:**
  Un servidor (físico o virtual) con un sistema operativo soportado (por ejemplo, Ubuntu 20.04 LTS o superior).
- **Acceso:**
  Credenciales de SSH para acceder de forma segura al servidor. Además, se debe disponer de un usuario con permisos suficientes (se recomienda no usar `root` directamente).
- **Dependencias Instaladas:**
  - Git
  - Node.js y npm (o yarn)
  - (Opcional) Docker y Docker Compose si se opta por una estrategia de contenedores
  - Nginx o Apache (para configurar un reverse proxy, si es necesario)
- **Otros:**
  Acceso al repositorio de *Namitask* y, si aplica, dominio o subdominio configurado para el servicio.

3. Preparación del Entorno en el Servidor

1. **Actualización de Paquetes y Creación de Usuario:**
   Actualiza los paquetes del sistema y, si es necesario, crea un usuario dedicado para el despliegue:

2. sudo apt update && sudo apt upgrade -y sudo adduser deploy # Crea un usuario 'deploy' para administrar la aplicación sudo usermod -aG sudo deploy
   Instalación de Dependencias Básicas:
   Instala Git, Node.js y npm. Por ejemplo, en Ubuntu:
3. sudo apt install git curl build-essential -y curl -sL https://deb.nodesource.com/setup_16.x | sudo -E bash - sudo apt install nodejs -y **(Opcional) Instalación de Docker:**
   Si prefieres el despliegue con contenedores, instala Docker:

sudo apt remove docker docker-engine docker.io containerd runc -y sudo apt install docker.io -y sudo systemctl start docker sudo systemctl enable docker # Agrega al usuario 'deploy' al grupo docker para evitar usar sudo en cada comando sudo usermod -aG docker deploy

4. Obtención del Código Fuente y Configuración del Proyecto

1. **Clonación del Repositorio:**
   Ingresa con el usuario *deploy* y clona el repositorio:
2. git clone https://tu-repositorio.com/tu-proyecto.git cd tu-proyecto **Configuración de Variables de Entorno:**
   Crea un archivo `.env` (o configura las variables en otro archivo de configuración) con los parámetros necesarios, por ejemplo:

PORT=3000 NODE_ENV=production
DATABASE_URL=postgres://usuario:contraseña@host:puerto/basedatos
SECRET_KEY=valor_secreto

Ajusta estos valores según tu entorno remoto.

5. Instalación de Dependencias y Construcción del Proyecto

1. **Instalación de Dependencias:**
   Ejecuta:
2. npm install

   Construcción de la Aplicación:
   Si la aplicación requiere un proceso de build (por ejemplo, para optimizar el front-end):

npm run build

Si usas un framework que maneja ambos entornos (front-end/back-end) en producción, asegúrate de revisar la documentación interna para pasos específicos.

6. Estrategias de Despliegue

Opción A: Despliegue Directo con Node.js y Reverse Proxy

1. **Ejecución como Servicio con systemd:**
   Crea un archivo de servicio para administrar la aplicación:

2. sudo nano /etc/systemd/system/namitask.service
    1. Con el siguiente contenido:
    2. [Unit] Description=Namitask Application After=network.target [Service] User=deploy WorkingDirectory=/home/deploy/tu-proyecto ExecStart=/usr/bin/npm start Restart=always EnvironmentFile=/home/deploy/tu-proyecto/.env [Install] WantedBy=multi-user.target

        1. Recarga los demonios y activa el servicio:
        2. sudo systemctl daemon-reload sudo systemctl enable namitask sudo systemctl start namitask **Configuración de Nginx como Reverse Proxy:**
           Instala Nginx:
    3. sudo apt install nginx -y

       Crea un archivo de configuración en `/etc/nginx/sites-available/namitask` con el siguiente contenido:
3. server { listen 80; server_name tu-dominio.com; location / { proxy_pass http://localhost:3000; proxy_http_version 1.1; proxy_set_header Upgrade $http_upgrade; proxy_set_header Connection 'upgrade'; proxy_set_header Host $host; proxy_cache_bypass $http_upgrade; } } Activa la configuración y reinicia Nginx:

sudo ln -s /etc/nginx/sites-available/namitask /etc/nginx/sites-enabled/ sudo nginx -t sudo systemctl restart nginx

Opción B: Despliegue con Docker

1. **Construcción de la Imagen Docker:**
   Asegúrate de tener un `Dockerfile` en el proyecto. Un ejemplo básico:
2. FROM node:16-alpine WORKDIR /app COPY package*.json ./ RUN npm install COPY . . RUN npm run build EXPOSE 3000 CMD ["npm", "start"]
    1. Construye la imagen:
    2. docker build -t namitask:latest .

       Ejecución del Contenedor:
       Ejecuta el contenedor, mapeando el puerto correspondiente:
    3. docker run -d --name namitask_container -p 3000:3000 --env-file .env namitask:latest

       (Opcional) Orquestación con Docker Compose:
       Si deseas administrar múltiples servicios (por ejemplo, una base de datos y la aplicación), crea un archivo `docker-compose.yml`:
3. version: '3' services: app: build: . ports: - "3000:3000" env_file: - .env restart: always # Agrega otros servicios, por ejemplo, una base de datos: db: image: postgres:13-alpine environment: POSTGRES_USER: usuario POSTGRES_PASSWORD: contraseña POSTGRES_DB: basedatos volumes: -

db-data:/var/lib/postgresql/data restart: always volumes: db-data:
Luego despliega con:

docker-compose up -d

7. Verificación del Despliegue

Una vez ejecutado el despliegue, realiza las siguientes acciones para asegurar que todo funcione correctamente:

- **Revisión de Logs:**
  Si usas systemd:
- sudo journalctl -u namitask -f
  - O, en el caso de Docker:
  - docker logs -f namitask_container

    Pruebas Manuales:
    Accede al dominio o dirección IP del servidor en un navegador y verifica que la aplicación carga correctamente y que las funcionalidades críticas operan sin inconvenientes.
  - **Monitoreo y Alerta:**
    Considera integrar herramientas de monitoreo (por ejemplo, Prometheus, Grafana o soluciones propias del proveedor en la nube) para estar al tanto del rendimiento y la estabilidad del servicio.
- 8. Solución de Problemas Comunes
  - **Error al Clonar o Actualizar el Repositorio:**
    Revisa las credenciales y la URL del repositorio.
  - **Errores en la Instalación o en el Build:**
    Confirma que todas las dependencias se instalaron correctamente y que la versión de Node.js es la requerida.
  - **Problemas de Conexión en Nginx o Reverse Proxy:**
    Verifica la configuración del proxy y asegúrate de que el puerto mapeado coincide con el configurado en la aplicación.
  - **Problemas de Permisos:**
    Asegúrate de que el usuario asignado (por ejemplo, *deploy*) tenga los permisos necesarios en las carpetas y archivos del proyecto.
- 9. Actualizaciones y Mantenimiento
  - **Actualización del Código:**
    Para actualizar la aplicación en el servidor, puedes hacer:
  - cd /home/deploy/tu-proyecto git pull origin main npm install # Solo si hay nuevas dependencias npm run build # Si es necesario sudo systemctl restart namitask

    Mantenimiento Regular:
    Realiza copias de seguridad de la base de datos y monitorea el estado del servicio para prevenir fallos. Documenta procedimientos de rollback en caso de errores críticos.
- 10. Recursos Adicionales

- **Documentación Oficial de Node.js:**
  Node.js Documentation
- **Guías de Docker y Docker Compose:**
  Docker Documentation
- **Referencia para Nginx:**
  Nginx Documentation
- **Uso de systemd:**
  systemd Manual