UNIVERSIDAD DEL VALLE DE GUATEMALA

Ingeniería de Software 1

Facultad de Ingeniería

Sección 10

LYNETTE GARCIA PÉREZ

Corte 3 del proyecto. Análisis y Diseño

Mario Rocha -23501
Joel Jaquez - 23369
Jonathan Zacarías - 231104
Luis González - 23353
Samuel Mejía - 23442

GUATEMALA, 10 de marzo 2025

Resumen:
FreelanceHub es una plataforma diseñada para conectar emprendedores con freelancers, facilitando la contratación de servicios y la gestión de proyectos de manera eficiente y segura. A lo largo del desarrollo del proyecto, se han identificado diversas necesidades clave tanto para emprendedores como para freelancers, lo que ha permitido iterar y mejorar las funcionalidades de la aplicación.

Durante el segundo corte, se enfocó en la captura de requisitos y la creación de prototípos de baja fidelidad, así como la validación con usuarios. En este tercer corte, se ha trabajado en nuevas iteraciones de los prototípos, implementación de funcionalidades solicitadas por los usuarios y refinamiento de los flujos de trabajo en la plataforma.

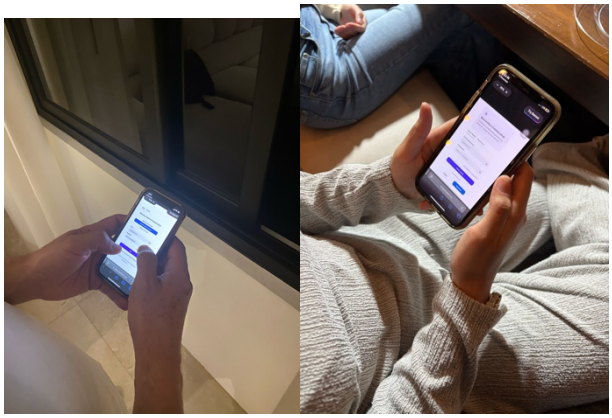
Entre las mejoras implementadas, se destaca la inclusión de pagos seguros, un sistema de contratos digitales, mejoras en la búsqueda de freelancers y la optimización del dashboard para la gestión de proyectos.

Introducción:
El auge del trabajo freelance y la necesidad de encontrar talento especializado han generado una creciente demanda de plataformas que faciliten la contratación de profesionales independientes. FreelanceHub surge como una solución a esta problemática, proporcionando un espacio donde emprendedores y freelancers puedan colaborar de manera eficiente, con garantías de seguridad en los pagos y una comunicación efectiva.

Desde el primer corte, el proyecto ha evolucionado significativamente. En la fase inicial, se realizó una investigación exhaustiva sobre las necesidades de los usuarios a través de entrevistas y encuestas. En el segundo corte, se trabajó en la creación de prototípos de baja fidelidad, así como en la definición detallada de requisitos. Ahora, en este tercer corte, se continúa refinando la solución a través de iteraciones basadas en la retroalimentación de usuarios y establecen las bases técnicas para su implementación.

En esta fase se han refinado los prototípos interactivos, implementando mejoras en los requisitos funcionales y no funcionales, y avanzado en la estructuración del diseño del sistema y arquitectura de los datos.

Desing Thinking:
Enlace al prototipo de los freelancers: https://app.banani.co/preview/AoNgitKMRTbhWhdLy8zR
Enlace al prototipo de los emprendedores: https://app.banani.co/preview/o1Nf3g4XLQEWUkTUrtBn



Durante la fase de investigación, se realizaron entrevistas a diferentes perfiles de usuarios como lo son:

- Freelancers
- Emprendedores
- Freelancers del interior
- Emprendedores multidisciplinarios

Se identificaron los siguientes puntos clave:

- Los emprendedores necesitan una plataforma confiable que garantice la calidad del servicio y la seguridad de los pagos.
- Los freelancers requieren un sistema que les brinde visibilidad y transparencia en los proyectos.
- La comunicación entre ambas partes es fundamental para el éxito de los proyectos.
- Se detectó la necesidad de un sistema de reputación y certificaciones para validar la experiencia de los freelancers.

Necesidades detectadas:

- Filtros avanzados para la búsqueda de freelancers y proyectos.
- Un sistema de pago seguro con opciones flexibles.
- Funcionalidad de mensajería para mejorar la comunicación
- Sección de contratos digitales para formalizar acuerdos.
- Historial de pagos y transacciones.

Iteraciones y Refinamiento del prototípo:

- Se agregó un sistema de pagos y contratos digitales.
- Se mejoró la interfaz de búsqueda con opciones más detalladas.
- Se optimizó la presentación de los perfiles de freelancers con certificaciones y proyectos destacados.
- Se implementó un sistema de notificaciones automáticas para pagos y actualizaciónes de proyectos.
- Se incluyó una mejor segmentación de proyectos dentro del dashboard.

Lista de cambios realizados en los prototipos:
Durante el desarrollo de los prototipos, se implementaron cambios basados en la retroalimentación de los usuarios entrevistados:

- Versión 1 (Prototipo Inicial - Baja Fidelidad)

-Estructura básica de la plataforma.

-Funcionalidades principales definidas.

-Diseño general de navegación y pantallas esenciales.

- Versión 2 (Prototipo de Alta Fidelidad - Primera Iteración)

  -Se mejoró la visibilidad de los filtros en la búsqueda de freelancers.

  -Se agregó la función de pagos por hitos y contratos digitales.

  -Implementación de un chat interno para facilitar la comunicación.

  -Se añadió la opción de destacar proyectos específicos en los perfiles de freelancers.

- Versión 3 (Prototipo de Alta Fidelidad - Segunda Iteración)

  -Se optimizó la interfaz de pagos y la visualización del historial de transacciones.

  -Se incorporó la opción de inicio de sesión con Google y Facebook.

  -Se mejoró la segmentación de los proyectos en categorías específicas.

  -Se añadieron notificaciones automáticas para fechas de entrega y pagos pendientes.

- Versión 4 (Prototipo de Alta Fidelidad - Versión Final)

  -Implementación de un sistema de insignias y niveles de experiencia para freelancers.

Se optimizó la presentación del perfil del freelancer con calificaciones y testimonios destacados.Se añadieron métricas de desempeño y recomendaciones automáticas de freelancers basadas en proyectos anteriores.

Cambios en los requisitos funcionales:

Tras cada iteración de los prototípos, se realizaron los siguentes cambios en los requisitos funcionales:

- Nuevo requisito: Implementación de un sistema de insignias y niveles para freelancers.
- Nuevo requisito: Agregar soporte para pagos fraccionados mediante hitos en los proyectos.
- Modificación: Se amplió la funcionalidad de búsqueda, permitiendo filtrar por nivel de experiencia y certificaciones.
- Modificación: Se añadió una sección de contratos digitales con firma electrónica.
- Nuevo requisito: Integración de notificaciones en tiempo real para cambios en los proyectos y pagos.

Estos cambios aseguran que la plataforma cumpla con las expectativas de los usuarios y proporcione una experiencia más optimizada y funcional.

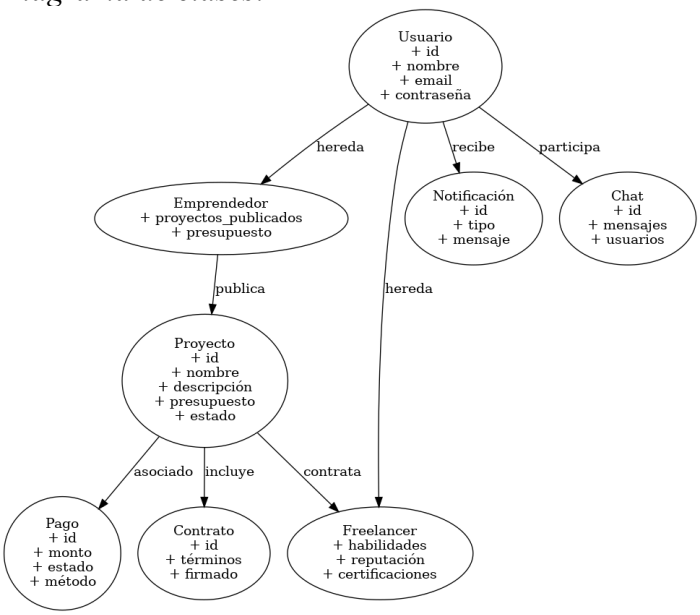
Análisis:

- **Lista de Requisitos Funcionales**

    - **Registro e Identificación de Usuarios**
      Historia de Usuario:
      Como usuario, quiero registrarme en la plataforma y seleccionar mi perfil (emprendedor o freelancer) para acceder a los servicios adecuados.
    - **Publicación de Proyectos**
      Historia de Usuario:
      Como emprendedor, quiero publicar proyectos con detalles claros para atraer freelancers adecuados.
    - **Búsqueda y Filtrado de Freelancers**
      Historia de Usuario:
      Como emprendedor, quiero buscar freelancers con filtros avanzados para encontrar el mejor candidato.
    - **Aplicación a Proyectos**
      Historia de Usuario:
      Como freelancer, quiero postularme a proyectos relevantes según mi experiencia y habilidades.
    - **Gestión de Pagos Seguros**
      Historia de Usuario:
      Como usuario, quiero procesar pagos de manera segura con garantía de entrega del trabajo.
    - **Contratos Digitales**
      Historia de Usuario:
      Como usuario, quiero establecer contratos digitales con términos claros para evitar malentendidos.
    - **Sistema de Notificaciones**
      Historia de Usuario:
      Como usuario, quiero recibir notificaciones sobre cambios en mis proyectos y pagos.
    - **Sistema de Calificación y Reseñas**
      Historia de Usuario:
      Como usuario, quiero ver calificaciones y reseñas de freelancers y emprendedores para tomar decisiones informadas.
    - **Chat Interno**
      Historia de Usuario:
      Como usuario, quiero una función de mensajería para comunicarme con mi equipo y clientes dentro de la plataforma.
    - **Historial de Pagos y Transacciones**
      Historia de Usuario:
      Como usuario, quiero acceder a un historial de pagos y transacciones para un mejor control financiero.
    - **Sistema de Insignias y Niveles**
      Historia de Usuario:
      Como freelancer, quiero obtener insignias y niveles según mi experiencia y reputación.

- *CLASES PRE ELIMINARES*

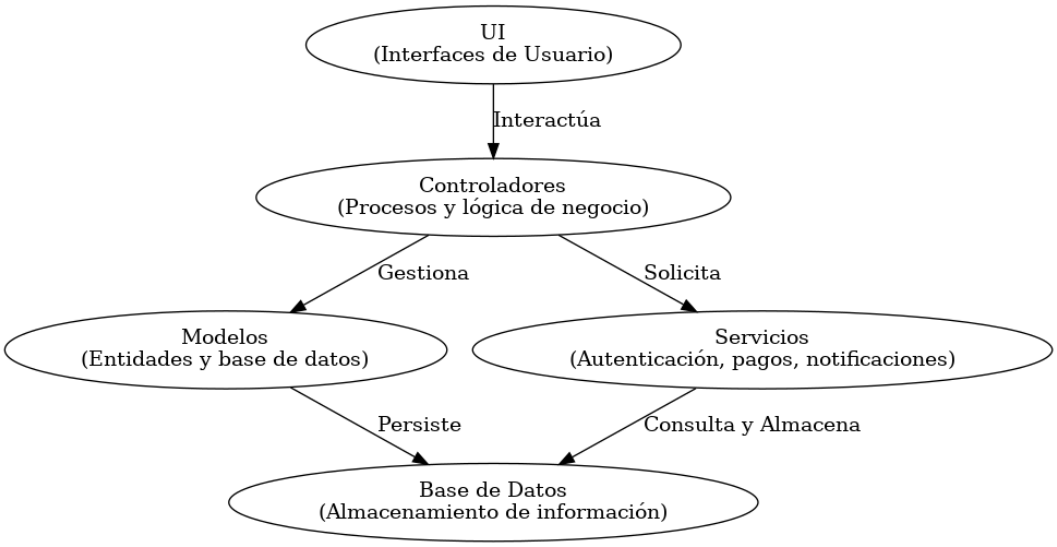
- *Diagrama de clases:*



- **Usuario**
- Atributos:
  - id: Identificador único del usuario.
  - nombre: Nombre del usuario.
  - email: Dirección de correo electrónico del usuario.
  - contraseña: Clave de acceso a la plataforma.
- Relaciones:
  - Clase padre de **Emprendedor** y **Freelancer**.
- **Emprendedor** *(Hereda de Usuario)*
- Atributos:
  - proyectos_publicados: Lista de proyectos publicados por el emprendedor.
  - presupuesto: Presupuesto disponible para contratación.
- Relaciones:
  - Puede **publicar** proyectos.
- **Freelancer** *(Hereda de Usuario)*
- Atributos:
  - habilidades: Lista de habilidades del freelancer.
  - reputación: Calificación basada en reseñas de clientes.
  - certificaciones: Certificaciones verificadas del freelancer.
- Relaciones:
  - Puede **ser contratado** en un proyecto.
- **Proyecto**
- Atributos:
  - id: Identificador único del proyecto.
  - nombre: Nombre del proyecto.
  - descripción: Detalles del proyecto.
  - presupuesto: Presupuesto asignado al proyecto.
  - estado: Estado del proyecto (abierto, en progreso, finalizado).
- Relaciones:
  - Publicado por un **Emprendedor**.
  - Puede **contratar** a un **Freelancer**.
- **Pago**

- Atributos:
  - id: Identificador único del pago.
  - monto: Cantidad de dinero transferida.
  - estado: Estado del pago (pendiente, completado).
  - método: Método de pago utilizado.
- Relaciones:
  - Asociado a un **Proyecto**.
- **Contrato**
- Atributos:
  - id: Identificador único del contrato.
  - términos: Condiciones establecidas entre emprendedor y freelancer.
  - firmado: Indica si el contrato ha sido firmado por ambas partes.
- Relaciones:
  - Incluido en un **Proyecto**.
- **Notificación**
- Atributos:
  - id: Identificador único de la notificación.
  - tipo: Tipo de notificación (mensaje, pago, alerta).
  - mensaje: Contenido de la notificación.
- Relaciones:
  - Enviada a un **Usuario**.
- **Chat**
- Atributos:
  - id: Identificador único del chat.
  - mensajes: Historial de mensajes intercambiados.
  - usuarios: Participantes en la conversación.
- Relaciones:
  - Usado para la comunicación entre **Usuarios**.

- Diagrama de paquetes:



- **Paquete UI (Interfaz de Usuario)**
  - **Descripción:** Contiene las interfaces gráficas y vistas que los usuarios interactúan dentro de la plataforma.
  - **Componentes:**

        - LoginView: Pantalla de inicio de sesión y registro de usuarios.
        - DashboardView: Vista principal donde los usuarios gestionan proyectos y conexiones.
        - ProjectView: Interfaz donde los emprendedores pueden publicar y administrar proyectos.
        - FreelancerProfileView: Página donde los freelancers configuran y gestionan su perfil.
        - ChatView: Interfaz de mensajería interna entre usuarios.
        - PaymentView: Pantalla de gestión de pagos y facturación.
- **Paquete Controladores (Lógica de Negocio y Procesos)**
    - **Descripción:** Contiene la lógica de negocio y se encarga de coordinar las interacciones entre la UI y los modelos de datos.
    - **Componentes:**
        - AuthController: Maneja la autenticación y autorización de usuarios.
        - ProjectController: Gestiona la creación, edición y eliminación de proyectos.
        - FreelancerController: Administra la información y reputación de los freelancers.
        - PaymentController: Procesa los pagos y garantiza su seguridad.
        - ContractController: Administra la generación y validación de contratos digitales.
        - NotificationController: Maneja el envío y gestión de notificaciones en la plataforma.
        - ChatController: Controla la mensajería interna entre usuarios.
- **Paquete Modelos (Entidades y Base de Datos)**
    - **Descripción:** Representa las entidades principales del sistema y define la estructura de la base de datos.
    - **Componentes:**
        - Usuario: Modelo base para los usuarios (emprendedores y freelancers).
        - Emprendedor: Extiende a Usuario, representando a los clientes que publican proyectos.
        - Freelancer: Extiende a Usuario, representando a los profesionales que aplican a los proyectos.
        - Proyecto: Modelo que representa los proyectos publicados.
        - Pago: Representa los pagos realizados dentro de la plataforma.
        - Contrato: Define los términos de contratación entre usuarios.
        - Notificación: Representa mensajes y alertas enviadas dentro de la plataforma.
        - Mensaje: Define la estructura de los mensajes intercambiados en los chats.

- **Paquete Modelos (Entidades y Base de Datos)**
  - **Descripción:** Contiene servicios adicionales y utilitarios que mejoran la funcionalidad del sistema.
  - **Componentes:**
    - AuthService: Gestiona la seguridad y autenticación de usuarios (incluye 2FA y OAuth).
    - PaymentService: Proporciona integraciones con pasarelas de pago (ejemplo: PayPal, Stripe).
    - EmailService: Maneja el envío de correos electrónicos transaccionales y notificaciones.
    - ChatService: Facilita la comunicación en tiempo real entre usuarios.
    - RecommendationService: Implementa algoritmos de recomendación para mejorar la conexión entre emprendedores y freelancers.
    - AnalyticsService: Proporciona métricas y reportes sobre el desempeño de los freelancers y proyectos.
- **Paquete Modelos (Entidades y Base de Datos)**
  - **Descripción:** Contiene las configuraciones y manejadores de bases de datos para el almacenamiento de la información.
  - **Componentes:**
    - DatabaseConfig: Configuración y conexión a la base de datos (ejemplo: PostgreSQL, MongoDB).
    - UserRepository: Accede y gestiona datos de usuarios en la base de datos.
    - ProjectRepository: Permite realizar consultas sobre los proyectos publicados.
    - PaymentRepository: Gestiona registros de pagos y transacciones.
    - ChatRepository: Maneja los mensajes intercambiados dentro de la plataforma.
    - NotificationRepository: Almacena y recupera las notificaciones enviadas a los usuarios.

Diseño:

Selección de la Tecnología de Desarrollo

Para garantizar el éxito de nuestra plataforma FreelanceHub, hemos realizado un análisis detallado de las opciones tecnológicas disponibles, tomando en cuenta los siguientes criterios fundamentales:

- **Escalabilidad:** La plataforma debe soportar hasta 10,000 usuarios simultáneos sin degradación del rendimiento.
- **Seguridad:** Se requiere autenticación segura (2FA), cifrado de datos y protección contra ataques informáticos.
- **Compatibilidad:** la aplicación debe ser accesible desde navegadores modernos como Chrome, Firefox, Edge y Safari, así como desde dispositivos móviles.
- **Tiempo de respuesta:** Se espera que las páginas carguen en menos de 3 segundo para una experiencia de usuario óptima.

Estimación del espacio por Usuario

Cada usuario podría generar datos en distintas secciones de la aplicación, como:

- Datos del perfil (nombre, email, contraseña cifrada, preferencias, etc.).
- Registros de actividad (logs, transacciones, interacciones).
- Datos adjuntos (imágenes, documentos, videos, etc.).

Si se realiza una estimación conservadora:

Tipos de Datos	**Espacio por Usuario**
Datos de perfil	5 KB
Registros de actividad (logs)	50 KB por mes
Datos transaccionales	100 KB
Documentos e imágenes	260 MB

Desglozando Documentos e imágens:

Tipo de archivo	**Peso Promedio**	**Cantidad Promedio Mensual**	**Total por Mes**
Imágenes (JPG, PNG)	1 MB c/u	50 imágenes	50 MB

Renders (JPEG, PNG, TIFF, EXR)	5 MB c/u	10 renders	50 MB
Videos (MP4, WebM, AVI)	10 MB por minuto	5 videos de 2 minutos	100 MB
Audios (MP3, WAV, OGG)	5 MB c/u	10 audios	50 MB
Otros documentos (PDF, DOCX, etc.)	2 MB c/u	5 documentos	10 MB

Total estimado por usuario:

$$0.005 + 0.3 + 0.1 + 2{,}000 = 2{,}000.405 \approx 2GB$$

Total para 10,000 usarios:

Si cada usuario necesita $\sim 2GB$

$$10{,}000 \times 2GB = 20{,}004.05\ GB \approx 20GB$$

Opciones Tecnológicas consideradas para Backend

Frameworks para Backend	Ventajas	Desventajas
Java con Spring Boot	• Alto rendimiento y escalabilidad. • Seguridad robusta con Spring Security. • Gran soporte empresarial y comunidad.	• Desarrollo más complejo y con mayor curva de aprendizaje. • Más consumo de recursos con tecnologías más ligeras.
Node.js con NestJS	• Ligero y rápido, ideal para aplicaciones en tiempo real. • Basado en JavaScript/TypeScript, fácil integración con frontend. • Alto rendimiento con arquitecturas escalables.	• No es la mejor opción para procesos pesado en CPU. • Requiere configuración adicional para garantizar seguridad robusta.
Django con Python	• Desarrollo rápido y seguro. • Buen manejo de bases de datos y ORM eficiente.	No tan eficiente en aplicaciones con alta concurrencia. Puede ser más lento en comparación con Node.js y Java.

	• Gran comunidad de soporte.	

Tecnología Seleccionada: Node.js con NestJS

Justificación

- **Alto rendimiento y escalabilidad:** Node.js es altamente eficiente en aplicaciones con muchas conexiones simultaneas, lo que es ideal para una plataforma que maneja interacciones en tiempo real.
- **Desarrollo modular y eficiente:** NestJS permite estructurar la aplicación en módulos, facilitando el mantenimiento y la escalabilidad futura.
- **Integración con tecnologías modernas:** Se alinea perfectamente con frameworks frontened como React y Vue.js.
- **Seguridad y autenticación robusta:** Se puede integrar con Passport.js para autenticación segura y gestionar OAuth2 o JWT.

Opciones Tecnológicas consideradas para Frontend:

Frameworks para frontend	Ventajas	Desventajas
React	- Ecosistema y comunidad enormes (múltiples librerías, foros de soporte y recursos). - Alto rendimiento con uso correcto de técnicas como lazy loading y memorización. - Flexibilidad: no impone arquitectura rígida, facilita implementar soluciones personalizadas. - Integración fluida con Node/NestJS (mismo lenguaje base, JavaScript).	- Puede volverse complejo en proyectos grandes sin una arquitectura clara. - Requiere aprender JSX y conceptos como el Virtual DOM y hooks, lo que puede ser algo confuso para nuevos desarrolladores.
Angular	-Framework completo con enrutador, inyección de dependencias y CLI integrados. - Soporte de Google y comunidad sólida. - Arquitectura y patrones bien establecidos, ideal para entornos empresariales.	- Curva de aprendizaje elevada: mayor “boilerplate” inicial y configuración. - Menos flexible en proyectos pequeños o de rápida iteración, ya que establece una estructura muy definida que puede sentirse pesada al inicio.
Vue	- Sintaxis amigable y curva de aprendizaje suave. - Reactividad nativa muy eficiente. - Fácil de integrar de forma progresiva en proyectos existentes o combinarlo con otros recursos.	- Comunidad un poco más pequeña que la de React o Angular (aunque en crecimiento constante). - Algunos plugins o librerías para funcionalidades específicas pueden no estar tan maduros o populares como en React.
Flutter	- Un solo código para web y móvil (Android/iOS).	- Ecosistema web menor que React o Angular (aunque

	- Interfaces muy ricas y potentes, con animaciones fluidas (similar a apps nativas). - Soporte de Google, creciente comunidad.	evoluciona rápido). - Tamaño del bundle inicial más grande por incluir el motor Flutter en la carga del navegador. - Lenguaje Dart requiere curva de aprendizaje adicional.

Frontend selecciónado: React

Justificación:

Tras comparar distintas opciones como Angular, Vue, Svelte y Flutter se decidió emplear React en JavaScript porque cuenta con un ecosistéma amplio y maduro, ofreciendo un alto rendimiento (posibilita tiempos de carga menores a 3 segundos si se aplican buenas prácticas), se integra naturalmente con Node.js en el backend y cumple con los principales requisitos no funcionales (escalabilidad, compatibilídad con navegadores modernos, seguridad y mantenibilidad). Esta elección garantiza que el proyecto pueda crecer sin problemas, aprovechar una extensa comunidad de desarrolladores y así satisfacer las expectativas de rendimiento y experiencia de usuario establecidas.

Selección de la Base de Datos

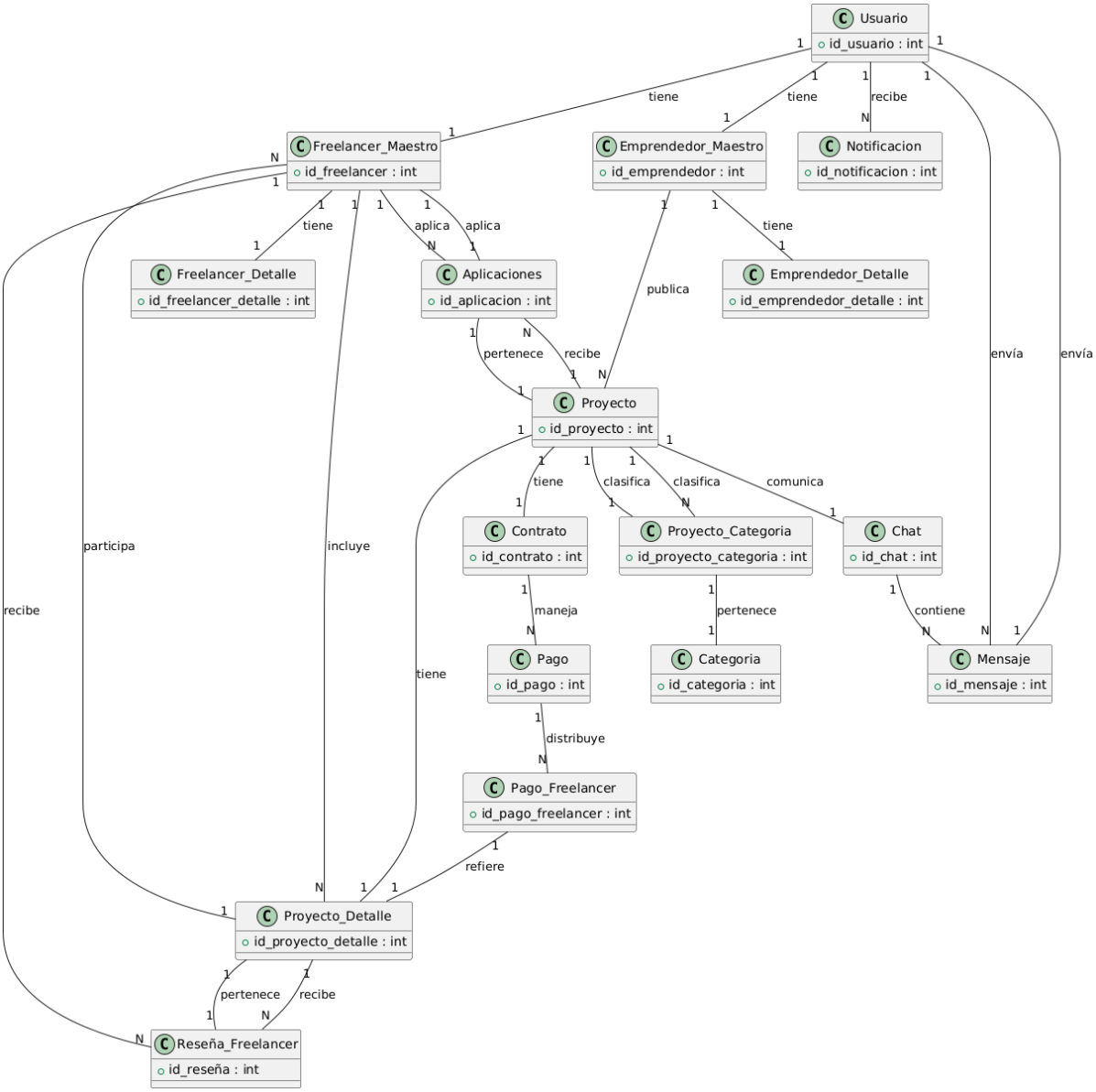
Base de datos	Ventajas	Desventajas
PostgreSQL	• Potente y escalable. • Sporte de transacciones y relaciones complejas. • Seguridad avanzada.	• Mayor consumo de recursos. • Puede requerir optimización en consultas de alto volumen.
MongoDB	• Flexible y escalable para grandes volúmenes de datos. • Esquema dinámico, ideal para manejar diferentes tipos de usuarios. • Fácil integración con Node.Js	• No es ideal para consultas complejas con muchas relaciones. • Puede consumir más espacio en disco que una base de datos relacional.
MySQL	• Gran compatibilidad y madurez. • Alto rendimiento en consultas relacionales.	• Menos eficiente que PostgreSQL para transacciones grandes. • No tan fexible como MongoDB para datos semi-estructurados.

Base de datos Seleccionada: PostgreSQL

Justificación

- **Seguridad avanzada y confiabilidad:** PostgreSQL ofrece soporte para cifrado y autenticación fuerte.
- **Manejo eficiente de datos estructurados:** Ideal para gestionar información crítica como usuarios, contratos, pagos y proyectos.
- **Soporte para transacciones y concurrencia:** Fundamental para los sistemas de pago seguro y la gestión de contratos dentro de la plataforma.

Diagrama de clases persistentes



Usuario: Clase base que puede ser un Freelancer_Maestro o un Emprendedor_Maestro.

Freelancer_Maestro: Contiene información básica del freelancer y se relaciona con Freelancer_Detalle, que almacena habilidades y certificaciones.

Emprendedor_Maestro: Representa a los emprendedores y se enlaza con Emprendedor_Detalle, que almacena información adicional como presupuesto y preferencias.

Proyecto: Un Emprendedor_Maestro puede publicar varios proyectos, y cada proyecto tiene un Proyecto_Detalle que define los freelancers involucrados.

Contrato y Pagos: Cada Proyecto tiene un Contrato, que a su vez puede generar varios Pagos distribuidos entre freelancers mediante Pago_Freelancer.

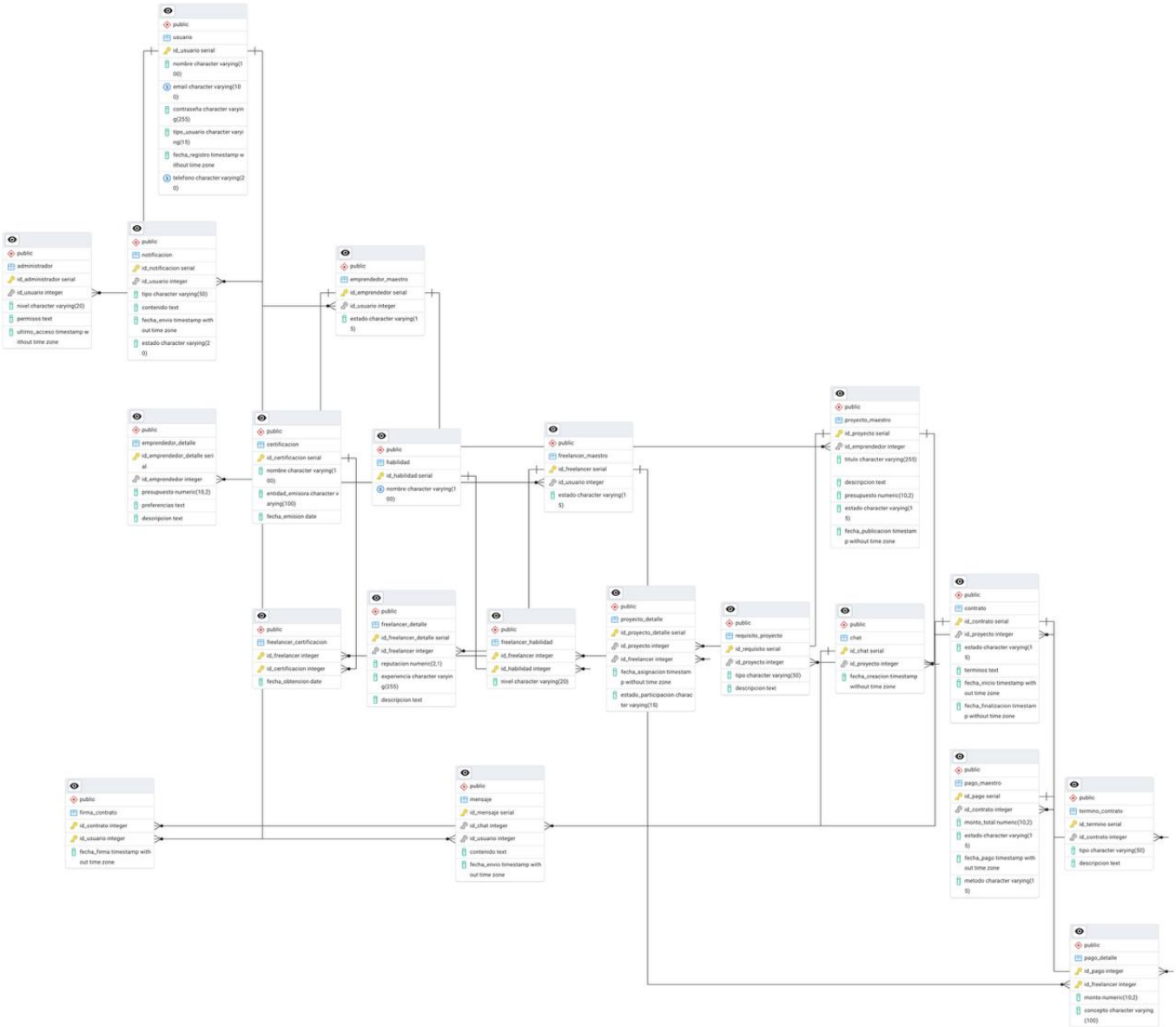
Aplicaciones y Reseñas: Los freelancers pueden aplicar a proyectos a través de Aplicaciones, y recibir evaluaciones mediante Reseña_Freelancer.

Comunicación: Cada Proyecto tiene un Chat para la comunicación entre freelancer y emprendedor, con Mensajes intercambiados por los usuarios.

Notificaciones: Los Usuarios reciben Notificaciones sobre eventos importantes en la plataforma.

Categorías: Los proyectos pueden clasificarse en diferentes Categorías a través de Proyecto_Categoria.

Diagrama Entidad-Relación



Enlaces

Enlace del Github:
https://github.com/jaq23369/FreelanceHub.git

Enlace Excel con la bitácora de trabajo:

https://uvggt-my.sharepoint.com/:x:/g/personal/roc23501_uvg_edu_gt/EdK4oXjb2hJOsMvaT_tHyd4BX-74K0e8staI9W3tIMaI1Q?e=5PBUoB