UNIVERSIDAD DEL VALLE DE GUATEMALA

Ingeniería de Software 1
Sección 20

Optimización de la gestión de inventario y planificación de materiales en Pool Center

Facultad de Ingeniería

Iris Ayala, 23965
Gabriel Bran, 23590
David Domínguez, 23712
Luis Padilla, 23663
Anggie Quezada, 23643

Guatemala

2025

Resumen

El presente trabajo se centra en la optimización de la gestión de inventario y planificación de materiales en Pool Center S.A., empresa especializada en el diseño y construcción de piscinas. Actualmente, enfrenta dificultades en el control de stock y asignación de materiales a proyectos, lo que afecta la eficiencia operativa.

Para abordar esta problemática, se desarrollará un sistema web que permitirá administrar el inventario, asignar materiales a proyectos y generar reportes. Este sistema incluirá módulos de autenticación de usuarios, control de stock con alertas de bajo inventario y un panel administrativo con reportes detallados. Además, contará con una interfaz pública para clientes y potenciales usuarios.

El objetivo principal del proyecto es mejorar la eficiencia en la gestión de recursos de Pool Center, permitiendo un seguimiento preciso del inventario y facilitando la planificación de materiales en proyectos. A través del desarrollo de una aplicación basada en Express.js, PostgreSQL y React.js, se busca una solución escalable, de fácil mantenimiento y adaptada a las necesidades operativas de la empresa.

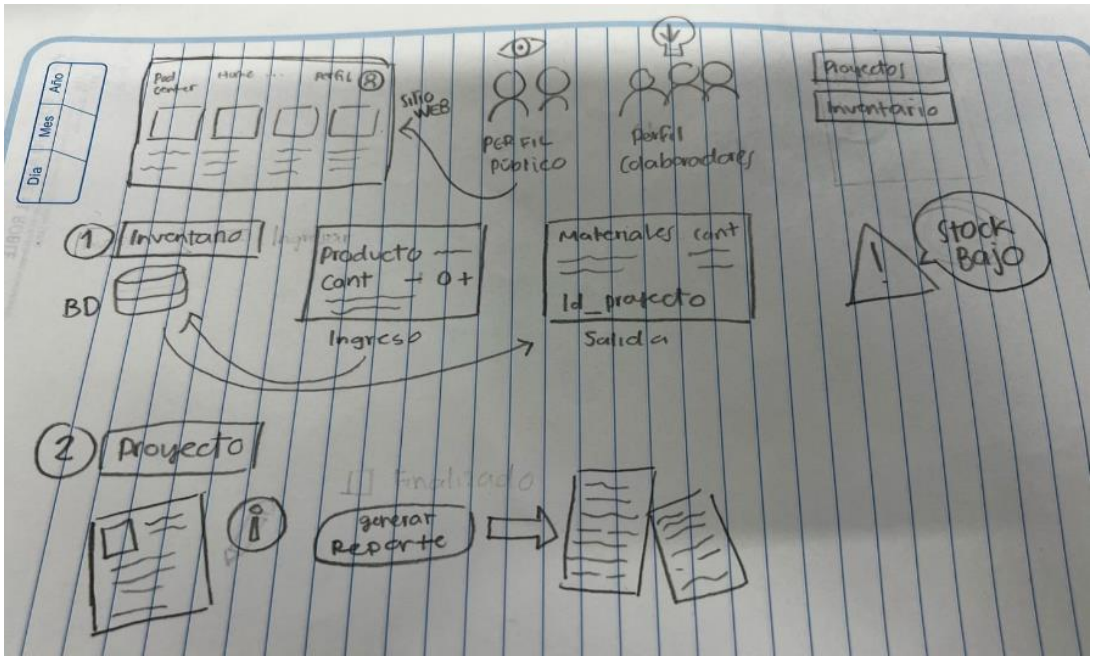
Introduccion

Pool Center S.A. es una empresa especializada en el diseño y construcción de piscinas. Actualmente, enfrenta dificultades en la gestión de su inventario, lo que genera problemas en la asignación de materiales y control de proyectos. Para solucionar esta problemática, se desarrollará un sistema web que permitirá administrar el inventario, asignar materiales a proyectos y generar reportes. La plataforma incluirá módulos para autenticación de usuarios, control de stock y notificaciones, asegurando una mejor organización y optimización de recursos. Este informe detalla el diseño y desarrollo del sistema, incluyendo el análisis de requisitos, la arquitectura propuesta y la selección de tecnologías

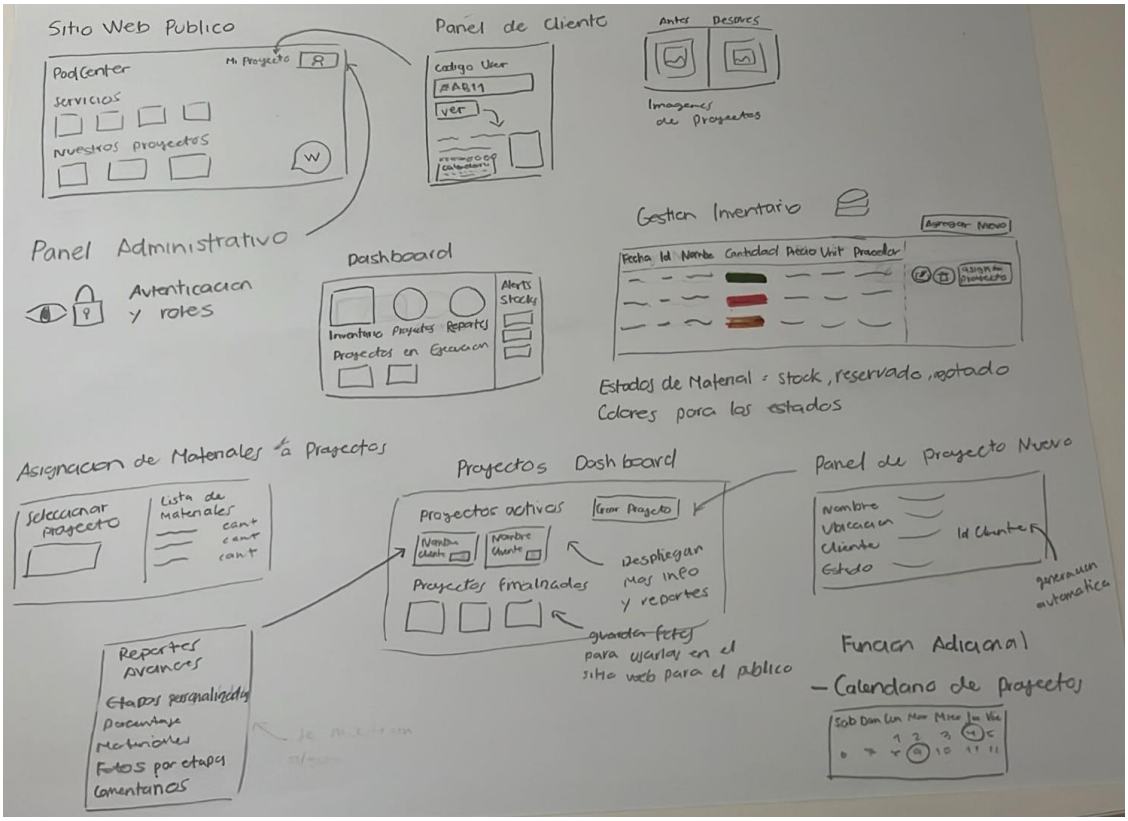
Design Thinking

Prototipos

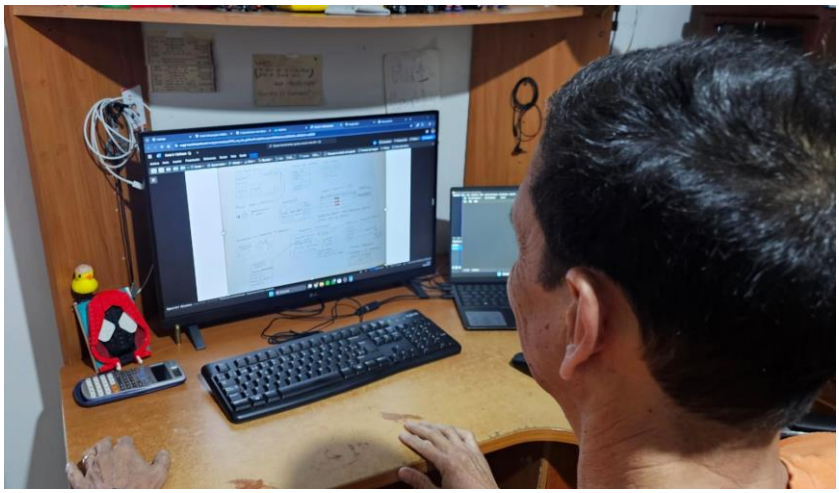
1. Consta en un sitio web donde como primera instancia se podrá observar la información de la empresa, como el tipo de servicios que ofrece, los diseños, la información de contacto. Pero también, mediante autenticaciones, los perfiles administrativos podrán acceder y tendrán acceso a información de los proyectos y del inventario. En la parte del inventario los que tengan permiso podrán ingresar un nuevo material y de la misma forma sacar materiales para asignárseles a un proyecto en específico, no solo sacar porque si, esto para llevar un control y que no se mezclen materiales entre proyectos. La plataforma también mandara mensajes de stock bajo cuando un material este por agotarse y de esa forma podrán cotizar los materiales con tiempo. En el apartado de proyectos, podrán poner toda la información y podrán generar reportes mensuales para ver los avances, retrasos, gastos, entre otros detalles acerca del proyecto.

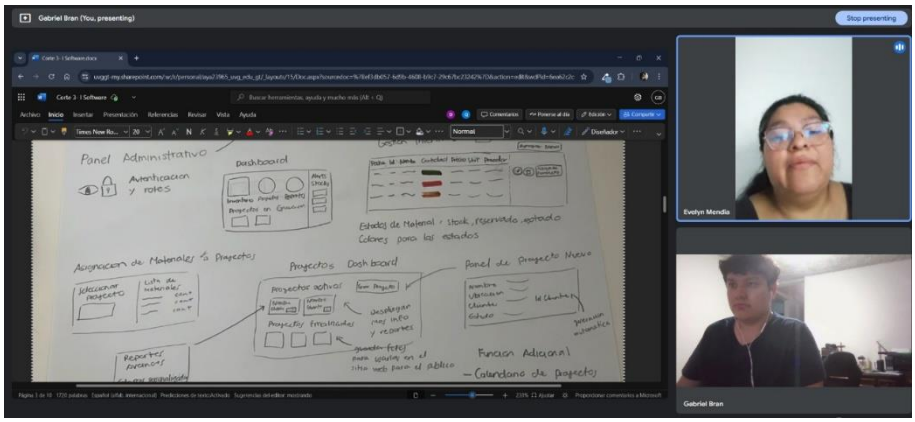


2. Similar al primer prototipo, este es un poco más elaborado en el tema de los módulos a incorporar en la aplicación web, sigue manteniendo la estructura de un módulo para el marketing y el contacto con la empresa y **se agregó** un panel para los clientes que ya tienen proyectos con la empresa. Cada cliente tiene un código y pueden ingresar ese código para ver los avances y calendarización de las actividades del proyecto. Y el otro modulo administrativo, en el que hay que **asegurar seguridad** y con la autenticación pueden ingresar a ese modulo. En el dashboard principal se cuenta con una vista de proyectos en ejecución, secciones para gestionar inventario, ver y crear proyectos, consultar reportes y un área especial para las notificaciones de alertas de stock del inventario. En el inventario se usará una base de datos relacional, y cada fila cuenta con botones de editar, eliminar o asignar a un proyecto. En la parte superior hay un botón para agregar un nuevo material. En este se usarán colores para hacer más visual el stock de los materiales (verde-stock, amarillo-bajo stock, rojo-agotado). De ahi cuenta con los diferentes dashboards de proyectos y reportes, para los proyectos se clasifican según el estado, se puede desplegar más información, los reportes cuentan con etapas personalizadas como (ej: Excavación, Estructuras, Revestimiento) y galería de fotos, se pensaba **agregar como funcionalidad un calendario** de las actividades, como (ej: Visita, Hormigón, Instalaciones, etc)



Interacción de los usuarios con los prototipos





Comentarios

Gerardo menciono directamente como quiere medir los materiales dentro del inventario. Nos mencionó que tiene mini inventarios por cada proyecto, por lo mismo le gusto que el dashboard tuviera los diferentes proyectos junto con cada uno teniendo lo que tiene el inventario apartado. De la misma manera le encanto que del inventario total se pueda sacar algo para destinarlo al proyecto que quisiera. Siguiente con esto dio que se tiene que poner por proyecto para la base de datos. En este siguiente ejemplo es para solo un proyecto:

Material	Ofrecido	Comprado	Obra	Bodega
Skimmer	6	4	0	2

Para cada proyecto le ofrece al cliente una cierta cantidad de material el cual compra en el momento, el que ya tiene en la obra y el que tiene en la bodega, ósea que la cantidad de material por obra no se tiene que pasar del ofrecido, pero por circunstancias Gerardo pide más inventario, así que ese exceso se tiene que ir a otra parte la cual se mantiene siempre vigente. También aclaro que quiere que para material dentro de cada proyecto se sepa la fecha de compra, la fecha en obra, el proveedor y la cantidad del material. Luego, le gusta la idea de que el cliente se mantenga actualizado con el update que se puede medir a través de un código dentro de la página web. Pero lo que no sabe es si se va a poder ya que las fotos que se toman son demasiado irregulares. Con los reportes mensuales lo que quieres son la información de compra, el inventario total de la bodega y los sobrantes de cada mes según los proyectos. Por último, la idea del calendario no le ve uso así que se puede quitar. Todo lo demás aprueba.

Ahora Evelyn no dio mucho más que se podría implementar más que información, como el tiempo que termina una obra para ver lo que se puede hacer del código del cliente para ver el progreso, también dijo que sería buena idea que dentro de la página donde se va a acceder, que se asegure de que no se puede ver a simple vista donde ingresen lo que van a ver el inventario. Luego a ella le siguen gustando la idea de las notificaciones de poco inventario.

Análisis

Requisitos funcionales

Módulo Público

- La página web debe mostrar información general de la empresa (servicios, fotos de proyecto, contacto).
- El sistema debe permitir visualizar cierta información de los proyectos. (Clientes)

Módulo de Autenticación

- El sistema debe permitir a los usuarios iniciar sesión con credenciales (usuario y contraseña).
- El sistema debe gestionar diferentes roles de usuario.
- El sistema debe permitir cerrar sesión a los usuarios autenticados.

Módulo de Inventario

- El sistema debe permitir registrar nuevos materiales con detalles completos.

- Los usuarios deben poder actualizar la cantidad de materiales disponibles al ingresar nuevas compras. (Gerente, Secretaría)
- Se debe permitir registrar la salida de materiales asignándolos a proyectos específicos. (Ingeniero, Gerente)
- El sistema debe generar alertas cuando los materiales alcancen o estén por debajo del punto de reorden.

Módulo de Proyectos

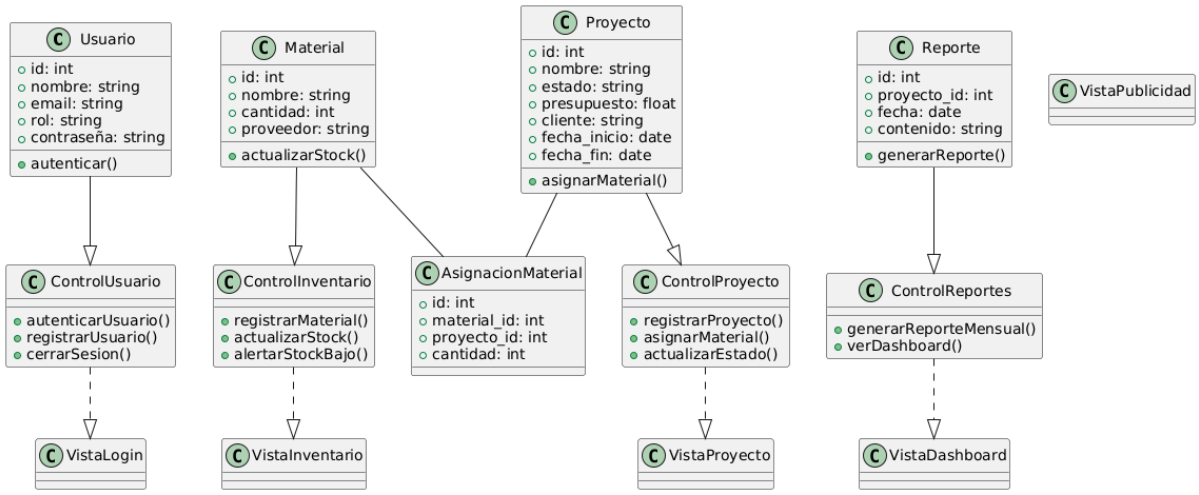
- Los usuarios deben poder registrar nuevos proyectos con detalles como nombre, ubicación, estado y presupuesto estimado, fecha, cliente. (Gerente, Ingeniero)
- Debe ser posible asignar materiales a proyectos específicos desde el inventario. (Ingeniero, Gerente)
- El sistema debe impedir la mezcla de materiales entre proyectos, exigiendo asignación específica.
- El sistema debe permitir actualizar el estado de un proyecto (en planificación, en ejecución, finalizado). (Ingeniero, Gerente)
- El sistema debe permitir subir y organizar fotos del avance por etapa.

Dashboard y Reportes

- El sistema debe mostrar un dashboard con resumen de proyectos activos e indicadores de estado.
- El sistema debe mostrar en el dashboard una lista de materiales con stock bajo.
- El sistema debe generar reportes mensuales de avances y materiales utilizados.

Clases preliminares

1. Diagrama de Clases y Descripción de las clases.



El sistema seguirá un patrón Modelo-Vista-Controlador (MVC)

Se dividirán las clases en:

- Modelo: Clases que representan entidades, es decir los datos y la estructura de la base de datos (Material, Proyecto, Usuario, etc.).
- Vista: Interfaces gráficas, Representan la UI en el frontend, interactuando con la API endpoints.
- Controlador: Clases que manejan la lógica del sistema (gestión de usuarios, inventario, proyectos) y las interacciones entre el modelo y la vista.

Modelo (Entidades):

- Usuario
    - Representa a los usuarios del sistema, con roles como gerente, ingeniero y secretaria.
    - atributos: id, nombre, rol, email, contraseña
- Material
    - Representa los materiales disponibles en inventario. Un material puede estar asignado a múltiples proyectos
    - atributos: id, nombre, cantidad, proveedor
- Proyecto
    - Contiene información sobre los proyectos en curso. Un proyecto tiene varios materiales asignados.
    - atributos: id, nombre, estado, presupuesto, cliente, fecha_inicio, fecha_fin
- AsignaciónMaterial
    - Relación intermedia entre Material y Proyecto, para controlar qué materiales pertenecen a cada proyecto.
    - atributos: id, material_id, proyecto_id, cantidad
- Reporte
    - Representa los reportes mensuales generados en el sistema.
    - atributos: id, proyecto_id, fecha, contenido

Vista (Interfaz):

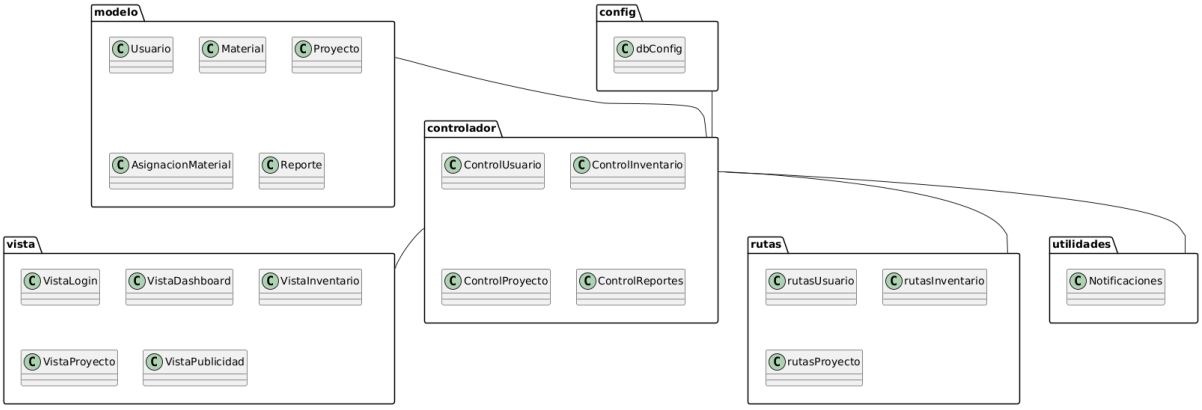
- VistaLogin
    - Muestra la pantalla de inicio de sesión y se conecta con ControlUsuario.
- VistaDashboard
    - Presenta los reportes y alertas en tiempo real.

- VistaInventario
    - Permite ver y administrar el inventario.
- VistaProyecto
    - Muestra los proyectos y su estado actual.
- VistaPublicidad
    - Muestra el apartado de conocer a la empresa.

Controlador (Controladora):

- ControlUsuario
    - Maneja la autenticación y gestión de usuarios.
    - métodos: autenticarUsuario(), registrarUsuario(), cerrarSesion()
- ControlInventario
    - Gestiona la entrada y salida de materiales.
    - métodos: registrarMaterial(), actualizarStock(), alertarStockBajo()
- ControlProyecto
    - Maneja la creación y asignación de proyectos.
    - métodos: registrarProyecto(), asignarMaterial(), actualizarEstado()
- ControlReportes
    - Gestiona la generación de reportes mensuales.
    - métodos: generarReporteMensual(), verDashboard()

2. Diagrama de paquetes y Descripción de cada paquete y de sus componentes



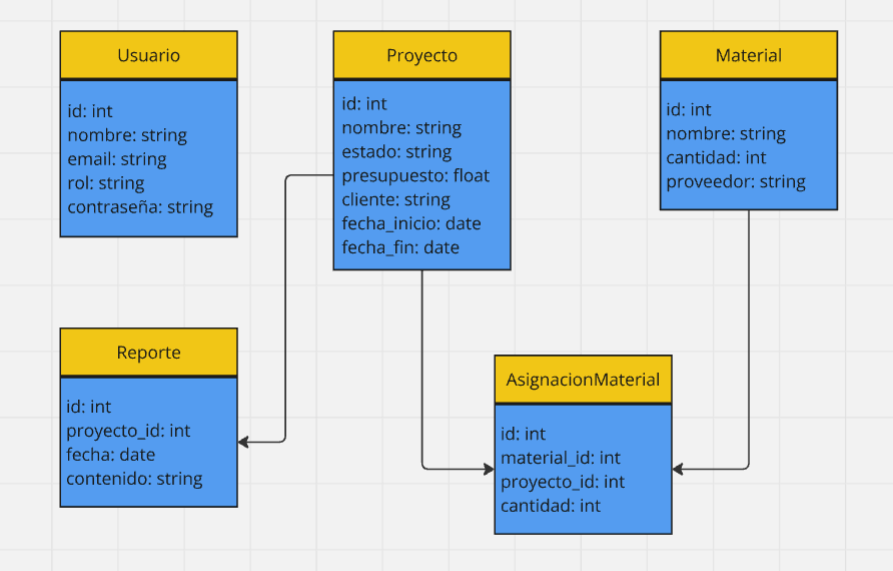
- modelo

    - Define las entidades y su relación con la base de datos.
    - Usuario, Material, Proyecto, AsignaciónMaterial, Reporte.
    - Por ejemplo: Material.js define la estructura de la tabla materiales.
- controlador
    - Maneja la lógica del negocio y la conexión entre el modelo y la API.
    - ControlUsuario, ControlInventario, ControlProyecto, ControlReportes.
    - Por ejemplo: ControlInventario.js se encarga de registrar materiales y gestionar stock.

- vista
    - Contiene los componentes que renderizan la interfaz de usuario.
    - VistaLogin, VistaDashboard, VistaInventario, VistaProyecto, VistaPublicidad.
    - Por ejemplo: VistaDashboard.js muestra reportes y alertas en tiempo real.
- rutas
    - Define los endpoints de la API para interactuar con el backend (Seguramente con Express.js).
    - rutasUsuario, rutasInventario, rutasProyecto, etc.
    - Por ejemplo: rutasProyecto.js maneja las solicitudes relacionadas con proyectos.
- utilidades
    - Maneja Clases auxiliares si son necesarias (ejemplo: Notificaciones).
- config
    - Contiene configuraciones globales del sistema.
    - Por ejemplo: dbConfig.js establece la conexión con PostgreSQL.

Persistencia

Diagrama de clases Persistentes



Descripción de tecnologías de almacenamiento

PostgreSQL:

- Integridad de los datos
  - PostgreSQL es un sistema de bases de datos transaccional que sigue estrictamente el modelo ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad). Esto garantiza que la información del inventario se mantenga coherente y segura, evitando inconsistencias en el almacenamiento y la asignación de materiales a los proyectos.
- Manejo de grandes volúmenes de datos
  - Dado que el sistema de inventario gestionará grandes volúmenes de información, como materiales, proyectos y asignaciones, es fundamental contar con una base de datos escalable. PostgreSQL permite manejar eficientemente grandes conjuntos de datos y, en caso de ser necesario, ampliar la estructura para incluir nuevas entidades sin comprometer el rendimiento.
- Costo
  - Al ser un software open source, PostgreSQL no genera costos de licencia, lo que lo convierte en una solución ideal para almacenar los datos de Pool Center sin incurrir en gastos adicionales.
- Seguridad
  - PostgreSQL ofrece mecanismos de seguridad que garantizan la protección de la información, como el cifrado de datos, tanto en tránsito como en almacenamiento, y la gestión de roles y permisos, que permite controlar el acceso a la base de datos según los privilegios de cada usuario.

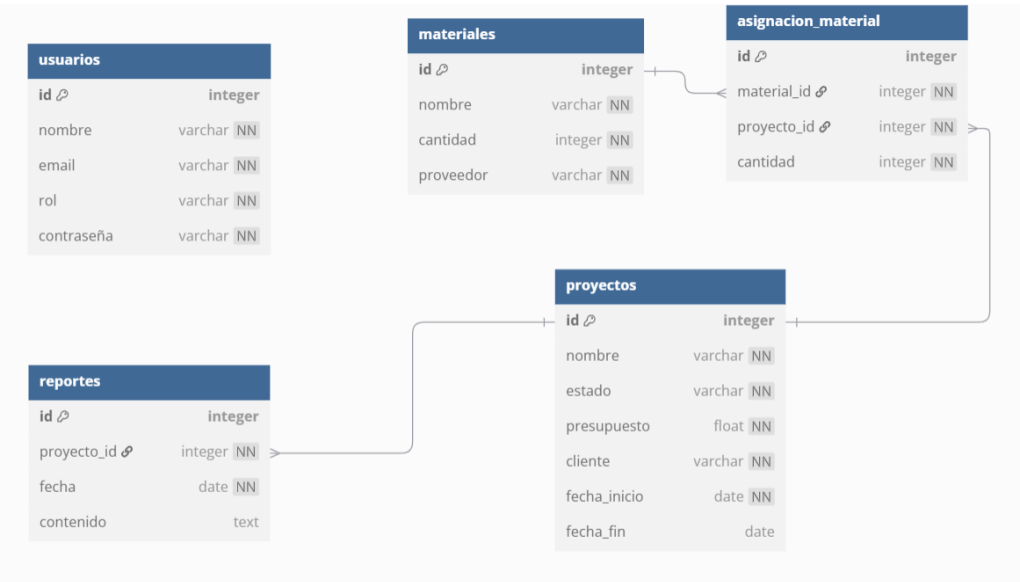
¿Por qué usar una base de datos relacional?

- Estructura organizada

- El uso de una base de datos relacional en un sistema de inventario es fundamental debido a su capacidad para organizar la información de manera estructurada. Permite gestionar datos sobre materiales, proyectos y asignaciones de materiales a proyectos con relaciones bien definidas, evitando la redundancia y asegurando un almacenamiento eficiente.

- Integridad y consistencia de datos
  - Las bases de datos relacionales cumplen con el modelo ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), lo que garantiza la integridad y coherencia de los datos. Esto es crucial para evitar errores en la asignación y modificación del stock, asegurando que todas las transacciones, como el ingreso de materiales al inventario, se realicen de manera segura y sin pérdidas de información.
- Consultas eficientes
  - Gracias al uso de SQL, las bases de datos relacionales permiten realizar consultas avanzadas de manera eficiente. Esto facilita la búsqueda rápida de materiales por nombre, proveedor o cantidad disponible, además de permitir la generación de reportes detallados sobre movimientos y niveles de inventario, lo que optimiza la toma de decisiones.
- Escalabilidad
  - Una base de datos relacional proporciona escalabilidad, permitiendo que el sistema crezca de manera controlada sin comprometer su estabilidad. A medida que aumenta el volumen de datos, la estructura relacional garantiza un rendimiento óptimo y facilita la expansión del inventario sin afectar la eficiencia del sistema.

Diagrama Entidad-Relación



Diseño

Análisis de Requisitos No Funcionales

Antes de seleccionar la arquitectura y tecnologías para el desarrollo del sistema, se establecieron los siguientes requisitos no funcionales para garantizar su rendimiento, seguridad y escalabilidad.

1. Cantidad de Usuarios

El sistema debe soportar entre 3 y 10 usuarios simultáneos en la fase inicial, con la posibilidad de escalar a un mayor número según el crecimiento de la empresa.

2. Tiempo de Respuesta

Las operaciones principales, como consultas de inventario y generación de reportes, deben ejecutarse en un máximo de 2 segundos bajo condiciones normales de carga. Para operaciones más complejas, el tiempo de respuesta no debe exceder los 5 segundos.

3. Escalabilidad

La arquitectura del sistema debe permitir la expansión sin comprometer su rendimiento. Debe poder soportar un aumento en la cantidad de usuarios, datos y operaciones, asegurando una respuesta eficiente a medida que la empresa crezca.

4. Tolerancia a Fallos y Disponibilidad

El sistema debe contar con mecanismos de respaldo y recuperación de datos para evitar pérdidas de información ante fallos del servidor. También debe permitir la recuperación del servicio en caso de interrupciones inesperadas.

5. Seguridad y Autenticación

El acceso a la plataforma debe estar restringido mediante un sistema de autenticación seguro. La información sensible debe estar protegida mediante buenas prácticas de seguridad, garantizando que solo usuarios autorizados puedan acceder a datos críticos

Selección de la Tecnología con base a los requisitos no funcionales

Frameworks para backend: Express.js, Laravel y Django

1. Express.js (Node.js)

- **Ventajas:**
  - Ligero y rápido, ideal para construir APIs REST eficientes.

    - Se programa en JavaScript, lo que permite usar el mismo lenguaje en frontend y backend.
    - Gran flexibilidad y control sobre la estructura del proyecto.
    - Comunidad enorme con muchas librerías disponibles.
    - Integración perfecta con frameworks modernos como React, Vue y Angular.
- **Desventajas:**
    - No incluye herramientas preconstruidas para autenticación o validaciones, se requiere más configuración manual.
    - Si no se organiza bien el código desde el inicio, puede volverse difícil de mantener en proyectos grandes.
    - La gestión de base de datos con PostgreSQL necesita un ORM externo como Sequelize o TypeORM.
- **Integración con frontend:**
    - Se integra fácilmente con React.js, Vue.js y Angular, permitiendo construir APIs optimizadas para aplicaciones SPA.

2. Laravel (PHP)

- **Ventajas:**
    - Framework completo con herramientas preconstruidas como autenticación, validaciones y ORM.
    - Eloquent ORM facilita la gestión de bases de datos como PostgreSQL.
    - Seguridad integrada con protección contra ataques comunes.
    - Buena estructura basada en MVC, lo que facilita el mantenimiento del código.
    - Comunidad grande y documentación extensa.
- **Desventajas:**
    - Más pesado y con mayor curva de aprendizaje que Express.js.
    - PHP no es la mejor opción si se busca escalabilidad en microservicios.
    - No está optimizado para APIs REST, aunque puede configurarse con Laravel API o Sanctum.
- **Integración con frontend:**
    - Se integra bien con Vue.js de forma nativa, pero también puede usarse con React.js o Angular con una configuración adicional.

3. Django (Python)

- **Ventajas:**
    - Seguridad robusta con protección contra ataques como CSRF, XSS y SQL Injection.
    - ORM poderoso que trabaja perfectamente con PostgreSQL.
    - Incluye un Admin Panel automático para la gestión de datos sin necesidad de crear una interfaz desde cero.
    - Buenas prácticas y estructura sólida que facilita el mantenimiento.
    - Comunidad activa con actualizaciones constantes.

- **Desventajas:**
  - Más complejo de aprender que Express.js debido a su estructura rígida.
  - No es nativamente asíncrono, lo que puede afectar el rendimiento en aplicaciones con muchas solicitudes simultáneas.
  - No está diseñado para APIs REST por defecto, por lo que se necesita Django REST Framework (DRF).
- **Integración con frontend:**
  - Funciona bien con React.js y Vue.js, pero requiere configurar Django REST Framework para consumir datos desde un frontend moderno.

Ganador: Express.js con Node.js

La mejor opción para este proyecto es Express.js porque es ligero, rápido y flexible, lo que permite construir una API eficiente para gestionar el inventario y los proyectos de Pool Center. Al estar basado en JavaScript, facilita la integración con React.js, Vue.js o Angular, lo que garantiza un desarrollo unificado entre el frontend y el backend. Además, su estructura minimalista permite optimizar el rendimiento sin sobrecargar el servidor con funcionalidades innecesarias. Aunque Laravel y Django ofrecen herramientas preconstruidas, estas pueden hacer que el sistema sea más pesado y menos adaptable a largo plazo. Express.js permite personalizar la arquitectura del backend de acuerdo con las necesidades del proyecto, asegurando una solución escalable y fácil de mantener.

Frameworks para fronted:

1. React.js

Ventajas:

- Biblioteca ligera y flexible, ideal para construir interfaces dinámicas y escalables.
- Uso de componentes reutilizables, lo que facilita la organización del código.
- Virtual DOM optimiza el rendimiento al minimizar cambios en el DOM real.
- Amplia comunidad y ecosistema, con muchas librerías y herramientas disponibles.
- Compatible con SSR (Server-Side Rendering) a través de Next.js para mejorar SEO y rendimiento.

Desventajas:

- No es un framework completo, por lo que requiere herramientas externas para gestión de estado (Redux, Context API).
- Puede ser más complejo de configurar en comparación con Vue.js.
- JSX puede resultar confuso para quienes no están acostumbrados a mezclar HTML con JavaScript.

Integración con backend:

- Se integra fácilmente con APIs REST y GraphQL.
- Funciona muy bien con Express.js, Django y Laravel.

- Puede consumir datos de backend a través de fetch o Axios.

2. Vue.js

Ventajas:

- Sintaxis sencilla y curva de aprendizaje baja, ideal para principiantes.
- Incluye herramientas integradas como Vue Router y Vuex para gestión de estado.
- Optimizado para rendimiento con su sistema de reactividad eficiente.
- Puede integrarse progresivamente en proyectos existentes sin necesidad de reescribir todo el código.
- Soporte nativo para animaciones y transiciones.

Desventajas:

- Comunidad más pequeña que React y Angular, lo que puede limitar algunas herramientas y recursos.
- No es tan utilizado en empresas grandes como React o Angular.
- Puede volverse complicado en proyectos muy grandes sin una estructura clara.

Integración con backend:

- Se integra bien con Laravel de manera nativa.
- Funciona con cualquier API REST o GraphQL, incluyendo Express.js y Django.
- Puede utilizar Axios o fetch para consumir datos del backend.

3. Angular

Ventajas:

- Framework completo con una estructura bien definida basada en TypeScript.
- Ofrece herramientas integradas como Angular CLI, Angular Router y RxJS para programación reactiva.
- Excelente rendimiento en aplicaciones grandes con su arquitectura modular.
- Soporte de Google y actualizaciones constantes.
- Ideal para aplicaciones empresariales con gran cantidad de datos y usuarios.

Desventajas:

- Curva de aprendizaje alta debido a TypeScript y su estructura rígida.
- Mayor cantidad de código en comparación con React y Vue.
- No es tan flexible como React para integrarse con otros proyectos.

Integración con backend:

- Se integra bien con APIs REST y GraphQL.
- Funciona con Express.js, Laravel y Django sin problemas.
- Usa HttpClient para realizar peticiones a servicios backend.

Ganador: React.js

La mejor opción para este proyecto es React.js debido a su flexibilidad, rendimiento y compatibilidad con múltiples backend como Express.js, Laravel y Django. Al ser una biblioteca ligera, permite una mayor personalización en la arquitectura de la aplicación sin imponer restricciones innecesarias. Además, su comunidad amplia garantiza soporte y nuevas herramientas para mejorar la experiencia de desarrollo. Vue.js es una excelente opción para proyectos más pequeños o cuando se busca facilidad de uso, mientras que Angular es ideal para aplicaciones empresariales con necesidades estructurales específicas.

Tecnologías para Persistencia:

1. PostgreSQL

Ventajas:

- Integridad de datos y cumplimiento ACID que Garantiza la consistencia y seguridad en transacciones críticas.
- Alta escalabilidad que Maneja grandes volúmenes de datos con eficiencia.
- Soporte para JSON y consultas avanzadas que permite combinar SQL con datos semiestructurados.
- Manejo de concurrencia avanzado: Usa MVCC (Multi-Version Concurrency Control) para reducir bloqueos.
- Extensibilidad la cual permite agregar funciones personalizadas y tipos de datos definidos por el usuario.

Desventajas:

- Mayor consumo de recursos: Requiere más memoria y CPU en comparación con alternativas más ligeras.
- Curva de aprendizaje moderada: Su configuración avanzada y optimización pueden requerir experiencia.

Integración con backend:

- PostgreSQL se integra fácilmente con Express.js mediante librerías como pg y ORM como Sequelize o TypeORM.
- Su modelo relacional permite manejar eficientemente el inventario y la asignación de materiales a proyectos.
- La escalabilidad de PostgreSQL lo hace ideal para soportar el crecimiento de la empresa.

2. MySQL

Ventajas:

- Rápido en consultas de solo lectura: Optimizado para operaciones de lectura en grandes volúmenes de datos.
- Amplia compatibilidad: Soportado por la mayoría de los servidores y herramientas.
- Fácil de aprender: Sintaxis y configuración más simples que PostgreSQL.
- Gran comunidad y documentación: Recursos abundantes para solución de problemas.

Desventajas:

- Gestión de concurrencia menos avanzada: Puede presentar bloqueos en transacciones concurrentes.
- Menos extensible: No permite tantos tipos de datos personalizados ni funciones avanzadas como PostgreSQL.
- Implementación parcial de ACID en algunos motores: InnoDB lo soporta, pero MyISAM no.

Integración con backend:

- Compatible con Express.js usando mysql2 o Sequelize.
- Puede manejar la base de datos de inventario y proyectos, aunque es menos flexible que PostgreSQL en operaciones complejas.
- Requiere ajustes adicionales para optimizar la concurrencia en múltiples usuarios.

3. SQLite

Ventajas:

- Ligero y sin necesidad de servidor: Ideal para aplicaciones pequeñas o embebidas.
- Fácil de configurar y utilizar: No requiere instalación ni administración compleja.
- Rápido para consultas simples: Excelente para bases de datos de tamaño reducido.

Desventajas:

- No adecuado para múltiples usuarios simultáneos: Soporta concurrencia limitada debido a su modelo de bloqueo de archivos.
- Menos eficiente en grandes volúmenes de datos: No es óptimo para sistemas con miles de registros y transacciones frecuentes.
- Sin soporte nativo para funciones avanzadas: Carece de características como permisos avanzados, replicación y concurrencia eficiente.

Integración con backend:

- Puede usarse con Express.js mediante sqlite3 o better-sqlite3, pero no es ideal para una aplicación web en producción.
- Podría ser útil para almacenamiento en dispositivos móviles o pruebas locales, pero no para un sistema de inventario en crecimiento.

Ganador: PostgreSQL

PostgreSQL es la mejor opción para la gestión de inventario de Pool Center S.A. debido a su fiabilidad, escalabilidad y compatibilidad con el stack tecnológico seleccionado. Su cumplimiento del modelo ACID garantiza transacciones seguras, mientras que su manejo de concurrencia mediante MVCC permite múltiples accesos sin afectar el rendimiento. Además, su soporte para consultas avanzadas y datos complejos facilita la generación de reportes

detallados. Se integra eficientemente con Express.js mediante ORM como Sequelize, y al ser open source, no genera costos de licencia.

Para el desarrollo del sistema de gestión de inventario y planificación de materiales de Pool Center, se ha seleccionado un stack tecnológico basado en Express.js para el backend, PostgreSQL para la base de datos y React.js para el frontend. Esta selección se realizó considerando los requisitos no funcionales del sistema, los cuales incluyen escalabilidad, rendimiento, confiabilidad y facilidad de mantenimiento.

Informe de gestión de tiempo

Integrante	Tarea	Fecha Inicio	Fecha Fin
Iris	Realizar Prototipos	14/3/2025	14/3/2025
Bran	Mostrar prototipos y colocar sugerencias y comentarios	15/3/2025	16/3/2025
Iris	Lista de requisitos funcionales	14/3/2025	14/3/2025
David	Clases preliminares	15/3/2025	16/3/2025
Luis	Persistencia de datos	16/3/2025	17/3/2025
Anggie	Comparacion y tecnologia seleccionada para el fronted. Requisitos no funcionales	15/3/2025	18/3/2025
Iris	Comparacion y tecnologia seleccionada para el backend	15/3/2025	18/3/2025
Bran	Comparacion y tecnologia seleccionada para persistencia	15/3/2025	18/3/2025
Bran	Introduccion	17/3/2025	19/3/2025
Iris	Resumen	17/3/2025	19/3/2025

Link del excel: Gestion de tiempos.xlsx

Link del historial de docs: Corte 3- I Software

Link del repositorio: https://github.com/Branuvg/IS_PoolCenter.git