PROYECTO DEL CURSO

Docente: Dr. Eric Gustavo Coronel Castillo

PROYECTO 1 Sistema de gestión de inventario para tienda minorista 2

PROYECTO 2 Sistema de gestión de citas médicas para una clínica 4

PROYECTO 3 Sistema de gestión de calificaciones escolares..... 7

PROYECTO 4 Sistema de seguimiento de entregas para mensajería local 10

PROYECTO 5 Sistema de reserva de entradas para un cine 13

PROYECTO 6 Sistema de gestión de cultivos para una granja 16

PROYECTO 7 Sistema de reservas para agencia de turismo..... 19

PROYECTO 8 Sistema de control de pedidos en un taller de manufactura..... 22

PROYECTO 9 Sistema de gestión de presupuesto personal..... 25

PROYECTO 10 Sistema de gestión de gimnasio y suscripciones 28

PROYECTO 11 Sistema de pedidos y reservas para restaurante 31

PROYECTO 12 Sistema de gestión inmobiliaria (propiedades y clientes) 34

PROYECTO 13 Sistema de gestión de biblioteca pública..... 37

PROYECTO 14 Sistema de gestión de voluntarios para una ONG..... 40

PROYECTO 15 Sistema de gestión de tickets de soporte técnico (Mesa de ayuda) 43

PROYECTO 16 Sistema Integral de Internamiento y Altas de Pacientes de una Clínica Privada 46

PROYECTO 17 Sistema de Gestión de Préstamos Personales para una Institución Financiera..... 49

PROYECTO 1
Sistema de gestión de inventario para tienda minorista

Descripción

La empresa “La Esquina” es una tienda minorista de barrio que vende productos variados. Actualmente llevan el control de inventario de forma manual en hojas de cálculo. Esta situación provoca errores de stock, dificulta saber qué productos reponer y ocasiona pérdidas de ventas cuando algún artículo se agota inadvertidamente. La tienda necesita un sistema de gestión de inventario que permita registrar y controlar las existencias de cada artículo en tiempo real, facilitando la toma de decisiones sobre compras y reposiciones.

Requerimientos funcionales

- **RF1:** El sistema deberá permitir registrar nuevos productos con detalles como nombre, código, categoría, precio y cantidad inicial en stock. Criterio de aceptación: Se considera cumplido si al ingresar los datos completos de un producto, este queda almacenado en el sistema y aparece en el listado de inventario disponible.
- **RF2:** Registrar ventas o salidas de stock reduciendo automáticamente la cantidad disponible del producto correspondiente. Criterio de aceptación: Al registrar una venta con el identificador del producto y la cantidad vendida, el sistema descuenta dicha cantidad del stock y refleja el nuevo nivel en la base de datos, sin permitir cantidades negativas.
- **RF3:** Generar alertas de reposición cuando el stock de un producto alcance un nivel mínimo predefinido. Criterio de aceptación: Si la cantidad en stock de un producto baja del umbral mínimo establecido (por ejemplo, 5 unidades), el sistema marca el producto con un estado de “bajo stock” y envía una notificación al administrador o muestra una alerta visible en la interfaz.
- **RF4:** Permitir la consulta del inventario por diferentes criterios, como por nombre de producto o categoría. Criterio de aceptación: Dada una palabra clave o categoría, el sistema muestra una lista filtrada de productos que coinciden, incluyendo sus cantidades disponibles, en un tiempo razonable (por ejemplo, menos de 2 segundos para una búsqueda típica).
- **RF5:** Generar reportes de inventario y ventas en un rango de fechas dado. Criterio de aceptación: Cuando el usuario solicita un reporte entre dos fechas, el sistema produce un informe que incluye el total de ventas por producto, ingresos totales y lista de productos agotados en ese período, presentándolo en pantalla o de forma descargable (p.ej., en PDF).

Requerimientos no funcionales

- **Usabilidad:** La interfaz debe ser intuitiva y fácil de usar, de modo que el personal de la tienda (con conocimientos técnicos mínimos) pueda aprender a operar el sistema en menos de un día.
- **Rendimiento:** Las operaciones comunes (como registrar una venta o buscar un producto) deben ejecutarse en pocos segundos (idealmente < 2 segundos) para no demorar la atención en caja.
- **Seguridad:** Solo el personal autorizado (p. ej., cajeros y gerente) podrá acceder al sistema mediante credenciales; los datos sensibles (como precios de costo) deben estar protegidos y no visibles para usuarios sin permiso.
- **Fiabilidad:** El sistema deberá asegurar la integridad de los datos de inventario, evitando pérdidas de información (por ejemplo, mediante respaldos diarios automáticos de la base de datos).
- **Escalabilidad:** La aplicación debe poder manejar un aumento en la cantidad de productos (por lo menos hasta ~10.000 ítems) sin degradación significativa del rendimiento, y permitir futuras modificaciones o ampliaciones con un esfuerzo razonable.

Detalles adicionales

- **Tipo de usuario final:** Empleados de la tienda (cajeros, gerente) con conocimientos básicos de computación.
- **Entorno de uso:** Computadores de escritorio o portátiles en la tienda (posiblemente un terminal en caja y otro en la oficina administrativa).
- **Tecnologías sugeridas:** Aplicación web cliente-servidor (por ejemplo, backend Java/Node.js y frontend web en HTML/JS) con base de datos relacional (SQL) para gestionar los registros de inventario.
- **Restricciones de tiempo:** Proyecto planeado para desarrollo en un semestre académico (~4-5 meses) con entregas parciales (iteraciones) que permitan ver prototipos tempranos.
- **Consideraciones:** El alcance se limita a una única tienda para mantener la complejidad baja; no se contempla integración con sistemas contables externos (los datos de ventas se registran manualmente solo en este sistema de inventario).

PROYECTO 2

Sistema de gestión de citas médicas para una clínica

Descripción

La Clínica “Salud y Vida” es un centro médico de tamaño mediano que ofrece consultas en distintas especialidades. La recepción actualmente gestiona las citas de forma manual en un cuaderno. Este proceso ocasiona a menudo empates de horario, dificultades para reprogramar citas y poca visibilidad de la agenda de cada médico. La clínica necesita un sistema de gestión de citas médicas que digitalice la programación de consultas, evitando conflictos de horario y mejorando la comunicación con los pacientes.

Requerimientos funcionales

- RF1: Permitir registrar una nueva cita médica asignando paciente, médico, fecha y hora, verificando disponibilidad. Criterio de aceptación: Cuando el recepcionista ingresa los datos de una cita (paciente, médico, fecha y hora), el sistema comprueba que ese médico no tenga otra cita en ese horario; si está libre, guarda la nueva cita y la muestra en la agenda correspondiente.
- RF2: Posibilitar la cancelación o reprogramación de citas existentes. Criterio de aceptación: Dada una cita previamente agendada, el sistema permite al usuario de recepción cancelarla (eliminándola de la agenda) o cambiarla a otro horario disponible. Tras la operación, la agenda se actualiza y, de ser posible, se notifica el cambio al paciente.

- RF3: Gestionar la información básica de pacientes para las citas (registro de nombre, contacto, historial de citas previas). Criterio de aceptación: Al programar una cita, si se ingresa un paciente no existente, el sistema permite registrar sus datos básicos (nombre, teléfono, identificación); si el paciente ya existe, se asocia la cita a su registro y se puede consultar su historial de atenciones previas.
- RF4: Enviar recordatorios automatizados de citas a los pacientes. Criterio de aceptación: Para cada cita programada con al menos 24 horas de anticipación, el sistema genera automáticamente un recordatorio (por correo electrónico o SMS) al paciente con la fecha y hora de su consulta, confirmando que el mensaje se ha enviado exitosamente.
- RF5: Permitir a los médicos y personal autorizado visualizar su agenda de citas. Criterio de aceptación: Cuando un médico inicia sesión, el sistema muestra una vista de calendario o lista con todas sus citas del día (u otro rango seleccionado), incluyendo detalles del paciente, y esta información debe estar disponible en todo momento para consulta.

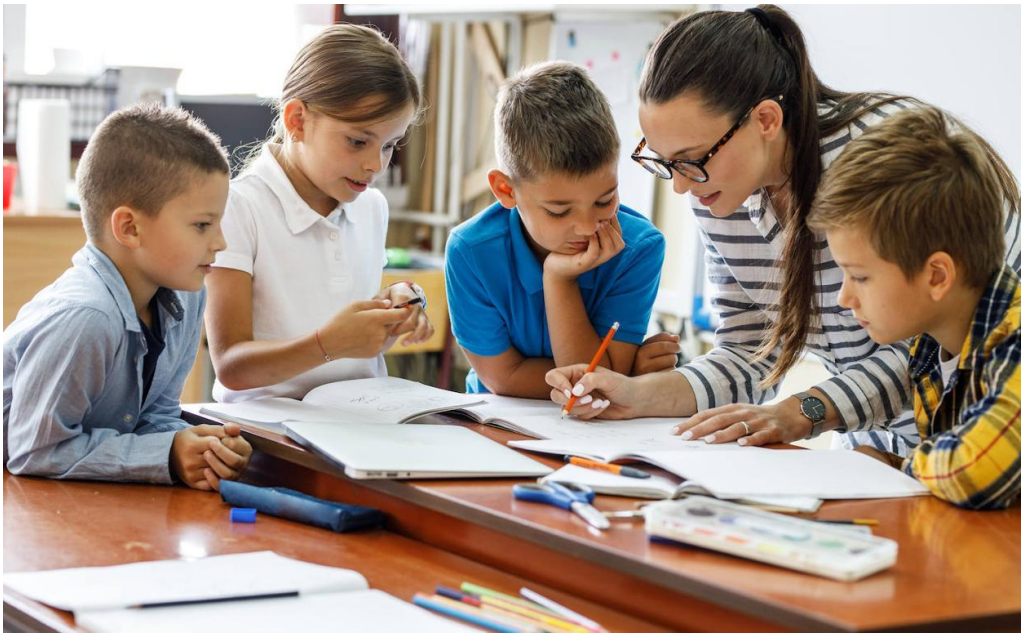
Requerimientos no funcionales

- Usabilidad: La aplicación debe ser fácil de usar para personal de recepción no técnico y médicos; la interfaz de programación de citas debe ser clara (por ejemplo, mostrar horarios ocupados vs disponibles de forma visual) para reducir errores.
- Eficiencia: Las operaciones de agendar, reprogramar o cancelar una cita deben efectuarse rápidamente (en < 2 segundos), para no hacer esperar a pacientes en mostrador o al teléfono.
- Seguridad y Privacidad: Los datos de los pacientes (información personal y registro de citas) deben almacenarse de forma segura y confidencial, cumpliendo normativas de privacidad (evitando accesos no autorizados mediante autenticación, y encriptando datos sensibles en la base de datos).
- Disponibilidad: El sistema debe estar disponible durante el horario de atención de la clínica (y permitir consultas fuera de ese horario a los médicos, si se habilita), con mínimas interrupciones, de modo que la agenda siempre pueda ser consultada o actualizada.
- Mantenibilidad/Escalabilidad: La solución debe poder adaptarse a futuros cambios, como agregar nuevas especialidades o más médicos, sin requerir reescrituras importantes. Un diseño modular y documentación facilitarán su mantenimiento y posibles ampliaciones.

Detalles adicionales

- Tipo de usuario final: Personal de recepción de la clínica y médicos (usuarios internos). Inicialmente los pacientes no interactúan directamente con el sistema (no hay portal de autoservicio para mantener la complejidad baja).
- Entorno de uso: Computadores de escritorio en recepción; los médicos podrían acceder al sistema desde sus consultorios mediante PC o tabletas dentro de la red de la clínica (vía aplicación web interna segura).
- Tecnologías sugeridas: Aplicación web o de escritorio en red local, con base de datos centralizada. Podrían emplearse frameworks web (como Django/Flask en Python, Java Spring, etc.) para facilitar la implementación. Integración con servicios de correo/SMS para los recordatorios (por ejemplo, a través de una API de email/SMS).
- Restricciones de tiempo: Proyecto acotado a un semestre académico (~4 meses); se recomienda desarrollar un prototipo funcional en iteraciones (primero agendamiento básico, luego agregando notificaciones y mejoras de interfaz).
- Consideraciones: El sistema se enfoca solo en la gestión de citas (no es una historia clínica electrónica completa). Se deben definir roles de usuario (ej.: recepcionista vs. médico) con diferentes permisos de acceso. La clínica modelo tiene ~5 médicos y espera manejar unas 50 citas por día, volumen que el sistema debe soportar cómodamente.

PROYECTO 3
Sistema de gestión de calificaciones escolares



Descripción

La institución educativa “Colegio ABC” enfrenta dificultades en el manejo de las calificaciones de sus estudiantes. Los profesores registran notas en cuadernos y luego las transfieren manualmente a planillas de Excel para generar reportes, proceso propenso a errores y que consume mucho tiempo durante la elaboración de boletines. Se necesita un sistema de gestión de calificaciones que centralice el registro de notas por curso y asignatura, permitiendo calcular promedios automáticamente y generar reportes de forma eficiente.

Requerimientos funcionales

- RF1: Permitir el registro de estudiantes, profesores y asignaturas, y la asociación de estudiantes a cursos/grados específicos. Criterio de aceptación: Se considera logrado si el administrador puede crear la lista de estudiantes con sus datos (nombre, ID, curso) y asignar cada estudiante a uno o más cursos/clases, así como registrar profesores y vincularlos a las asignaturas que impartirán.
- RF2: Permitir que los profesores ingresen calificaciones de evaluaciones (tareas, exámenes, etc.) para los estudiantes de sus clases. Criterio de aceptación: Cuando un profesor selecciona su clase y la evaluación correspondiente, el sistema presenta la lista de estudiantes; al introducir una calificación para cada estudiante y guardarla, los datos quedan almacenados y accesibles para futuras consultas o ediciones por el mismo profesor.

- RF3: Calcular automáticamente promedios parciales y finales por estudiante en cada asignatura. Criterio de aceptación: Dadas varias calificaciones ingresadas para un estudiante en una materia, el sistema calcula el promedio (aplicando los pesos o fórmulas definidas, por ejemplo, promedio aritmético simple) y muestra el resultado. El profesor puede verificar que el promedio corresponde a las notas ingresadas.
- RF4: Generar reportes de calificaciones, tales como boletines por estudiante o listados de notas por curso. Criterio de aceptación: Al solicitar un boletín de un estudiante, el sistema produce un documento o pantalla con las notas finales de ese alumno en cada asignatura del período, listo para impresión o exportación. De forma similar, un profesor puede obtener un listado de todas las notas de su clase en una evaluación determinada.
- RF5: Restringir el acceso y edición de notas según roles de usuario. Criterio de aceptación: Si un profesor inicia sesión, solo puede ver y modificar las calificaciones de las clases/asignaturas que tiene asignadas. Si un administrador inicia sesión, puede ver todas las clases y realizar acciones globales (como corregir datos de un estudiante). Ningún usuario puede alterar notas de clases ajenas a su rol, garantizando integridad y privacidad.

Requerimientos no funcionales

- Usabilidad: Interfaz amigable para docentes; por ejemplo, un formulario de ingreso de notas similar a una hoja de cálculo para que les resulte familiar, con validaciones claras (no permitir notas fuera de rango, etc.).
- Seguridad: Solo usuarios autenticados pueden acceder a las calificaciones. Se debe llevar un registro de cambios en las notas (bitácora) para asegurar trazabilidad y evitar manipulaciones indebidas.
- Integridad de datos: La aplicación debe evitar pérdidas de información, guardando automáticamente los datos ingresados y realizando copias de seguridad periódicas (por ejemplo, diaria o por período académico) para no perder historial en caso de falla.
- Rendimiento: El sistema debe manejar información de cientos de estudiantes y decenas de clases sin ralentizarse. Las consultas de notas y generación de reportes deben completarse en pocos segundos.
- Escalabilidad: Debería ser posible extender el sistema para múltiples ciclos escolares (mantener históricos de años anteriores) y eventualmente integrar un portal para estudiantes/padres, sin necesidad de reestructuraciones mayores.

Detalles adicionales

- Tipo de usuario final: Profesores (ingreso y consulta de notas de sus materias) y personal administrativo o de dirección del colegio (consulta global y administración de datos). Los estudiantes o padres no acceden directamente en esta versión.
- Entorno de uso: Computadores de escritorio o portátiles; puede funcionar en la red interna del colegio y/o vía Internet para permitir a los docentes ingresar notas desde casa mediante una interfaz web segura.
- Tecnologías sugeridas: Aplicación web con backend en un framework (por ejemplo, Laravel/PHP, Ruby on Rails, Django) y base de datos relacional para almacenar notas, estudiantes y cursos. Frontend con interfaz web simple para formularios de notas y tablas de resumen; se pueden usar librerías para gráficos simples en la visualización de estadísticas.
- Restricciones de tiempo: Desarrollo previsto en unas 12-16 semanas como proyecto académico. Se puede implementar por etapas: primero ingreso de notas y cálculo de promedios, luego reportes, después control de accesos por rol y finalmente refinamientos.
- Consideraciones: El sistema no aborda gestión de asistencia ni módulos más complejos de administración escolar; se enfoca únicamente en calificaciones. Se asume un colegio de tamaño mediano (~200 estudiantes, ~20 profesores) para dimensionar pruebas de rendimiento. Los aspectos de cálculo avanzado (ponderaciones complejas, curvas) se pueden simplificar para efectos del ejercicio.

PROYECTO 4
Sistema de seguimiento de entregas para mensajería local

Descripción

La empresa de mensajería “Envíos Express Locales” se dedica a recoger y entregar paquetes dentro de la ciudad. Actualmente manejan el seguimiento de entregas mediante llamadas telefónicas y registros en papel, lo que genera confusión acerca del estado de cada envío y dificulta informar a los clientes sobre sus entregas. Se propone desarrollar un sistema de seguimiento de entregas que permita registrar cada paquete, asignarlo a un repartidor y actualizar su estado en tiempo real, mejorando la eficiencia y la transparencia del servicio.

Requerimientos funcionales

- RF1: Registrar cada envío con sus datos (remitente, destinatario, dirección de entrega, descripción del paquete, etc.) y generar un identificador único de seguimiento. Criterio de aceptación: Al ingresar la información completa de un nuevo paquete en el sistema, este le asigna un número de seguimiento único y almacena el registro. El envío aparece en la lista de entregas pendientes con todos sus detalles.
- RF2: Permitir asignar cada paquete a un repartidor específico para su entrega. Criterio de aceptación: Dado un envío registrado, el usuario (coordinador de logística) puede seleccionar un repartidor de la lista y marcar el envío como “asignado”. El sistema registra el repartidor asociado al envío y actualiza el estado a “En ruta” (u otro indicador de que está en proceso de entrega).
- RF3: Actualizar el estado de un envío según su progreso (por ejemplo: “Pendiente de recogida”, “En ruta”, “Entregado”, “Incidencia/No entregado”). Criterio de aceptación: El repartidor o personal de oficina puede cambiar el estado de un envío. Por ejemplo, al marcar “Entregado”, el sistema registra la hora de entrega, el estado final, y queda reflejado en la base de datos y en la consulta de ese envío.
- RF4: Permitir la consulta del estado actual de un envío mediante el identificador de seguimiento. Criterio de aceptación: Si se ingresa el número de seguimiento de un paquete, el sistema muestra su información básica (destinatario, dirección) y el estado actualizado (etapa en la que se encuentra o si ya fue entregado) de forma inmediata, facilitando responder consultas de clientes.
- RF5: Notificar la entrega exitosa o incidencias relevantes. Criterio de aceptación: Cuando un envío cambia a estado “Entregado”, el sistema genera

automáticamente una notificación (por correo electrónico o SMS) al remitente y/o destinatario informando que el paquete fue entregado con éxito (o registra un mensaje de incidencia si no pudo entregarse), confirmando que la información llegó a los contactos designados.

Requerimientos no funcionales

- Usabilidad: La interfaz debe ser sencilla para el personal de logística. Por ejemplo, presentar un panel con listados de envíos pendientes y entregados, y botones claros para actualizar estados. Si los repartidores usan el sistema, la interfaz (posiblemente móvil) debe ser muy simple (botones grandes para cambiar estado) dado que la usarán en campo.
- Rendimiento: Las actualizaciones de estado deben reflejarse casi en tiempo real (< 1–2 segundos) para que la información esté siempre actualizada. El sistema debe soportar decenas de entregas diarias sin degradación (ej., al menos 100 envíos por día).
- Confiabilidad: El sistema debe ser estable durante las horas de operación (aprox. 8am–8pm) y asegurar que no se pierdan registros de envíos. Debe incluir mecanismos de respaldo de datos (backup diario de la base de datos) para recuperar información en caso de falla.
- Seguridad: La información de envíos (direcciones, datos de clientes) debe estar protegida; solo el personal autorizado y los repartidores pueden acceder. Se requiere autenticación para ingresar al sistema y se debe registrar quién realiza cambios de estado en cada entrega (trazabilidad).
- Escalabilidad: Aunque inicia con operaciones locales, el diseño debe contemplar la posibilidad de crecer en volumen (por ejemplo, manejar varios cientos de entregas diarias) o incorporar nuevas funcionalidades (p.ej., seguimiento GPS en el futuro) sin requerir rehacer el sistema por completo.

Detalles adicionales

- Tipo de usuario final: Coordinador de entregas y personal de oficina (para registrar y monitorear envíos) y repartidores (para actualizar el estado, si se les provee acceso). Los clientes reciben información, pero no acceden al sistema directamente.
- Entorno de uso: Oficina central con computadoras de escritorio para la gestión; potencial uso de smartphones por parte de repartidores en campo para actualizaciones (requiere conexión de datos móvil). Posible arquitectura híbrida: interfaz web para oficina y aplicación móvil ligera para repartidores.
- Tecnologías sugeridas: Aplicación web responsiva (usable desde PC y dispositivos móviles). Backend con API REST para actualizar estados (por

ejemplo, Node.js/Express o Java Spring) y base de datos SQL. Integración con servicios de correo/SMS para notificaciones (por ejemplo, Twilio para SMS, SMTP para email).

- Restricciones de tiempo: Proyecto realizable en un curso semestral. Se puede iniciar con un prototipo enfocado en registro de envíos y actualización de estados, y luego agregar características como notificaciones y roles de usuario.
- Consideraciones: Se asumirá cobertura de red móvil suficiente para permitir actualizaciones en tiempo real por los repartidores. El sistema se enfocará en el seguimiento básico de paquetes; optimizaciones de rutas, integración de mapas o geolocalización quedan fuera de alcance para mantener la simplicidad.

PROYECTO 5
Sistema de reserva de entradas para un cine

Descripción

La empresa “CineStar Barrio” es un cine local con varias salas que vende entradas en taquilla y por teléfono. Actualmente, las reservas telefónicas se anotan en una libreta y no siempre se sincronizan bien con la venta en taquilla, lo que ocasiona a veces sobreventa de asientos o confusiones con las reservas. Se busca implementar un sistema de reserva de entradas que centralice la información de asientos disponibles por función (horario de película), permitiendo gestionar en tiempo real la venta y reserva de boletos para evitar duplicidades y mejorar la experiencia de los clientes.

Requerimientos funcionales

- RF1: Gestionar la cartelera de películas y funciones (horarios) del cine. Criterio de aceptación: El administrador puede registrar nuevas películas con sus horarios y sala asignada. Al guardar, las funciones aparecen en el sistema con sus detalles (película, sala, hora, aforo de asientos) listas para recibir reservas.
- RF2: Permitir la selección y reserva de asientos para una función específica. Criterio de aceptación: Cuando un cliente (o empleado en taquilla) selecciona una función y elige un asiento específico en el mapa de la sala, el sistema marca ese asiento como reservado/ocupado y registra la reserva a nombre del cliente. No se debe poder elegir el mismo asiento dos veces para la misma función por distintos usuarios.
- RF3: Registrar la venta o reserva con los detalles del cliente. Criterio de aceptación: Al confirmar una reserva/venta, el sistema almacena el nombre del cliente y opcionalmente un número de contacto; tras la confirmación, se genera un código o número de boleto único asociado a esa reserva, que queda guardado en la base de datos para su posterior verificación en la entrada a la sala.
- RF4: Evitar sobreventas y manejar el cupo de cada función. Criterio de aceptación: El sistema no permitirá confirmar una reserva si la sala ya alcanzó el límite de asientos vendidos/reservados para esa función. Si se intenta reservar cuando ya no quedan asientos disponibles, se mostrará un mensaje de error indicando que el cupo está lleno y la transacción no se procesará.
- RF5: Permitir cancelaciones o modificaciones de reservas por parte del personal autorizado. Criterio de aceptación: Si un cliente desea cancelar su reserva, un empleado puede localizar la reserva por código o nombre y

anularla; el asiento correspondiente vuelve a marcarse como disponible inmediatamente, permitiendo que otro cliente lo pueda reservar. La cancelación queda registrada (con hora y usuario que la realizó).

Requerimientos no funcionales

- Usabilidad: La interfaz de reserva debe ser clara y fácil de usar, presentando un mapa visual de asientos para cada sala. Tanto clientes en línea (si se habilita) como empleados de taquilla deben poder seleccionar asientos rápidamente sin capacitación extensa.
- Rendimiento: El sistema debe procesar la reserva de un asiento en tiempo real; incluso ante múltiples usuarios reservando simultáneamente, la actualización del estado de los asientos debe ocurrir en segundos. Debe soportar picos de uso (por ejemplo, durante estrenos en fin de semana) sin ralentizaciones notables.
- Concurrencia: Debe garantizar la consistencia al reservar asientos concurrentemente. Si dos usuarios intentan el mismo asiento a la vez, solo el primero en confirmar debe obtenerlo, evitando condiciones de carrera mediante mecanismos de bloqueo o transacciones atómicas en la base de datos.
- Disponibilidad: La aplicación debería estar disponible en línea 24/7 para permitir reservas en cualquier momento. Las ventanas de mantenimiento deben ser mínimas (por ejemplo, en horarios de madrugada cuando no hay funciones) para no interrumpir el servicio.
- Seguridad: Aunque inicialmente no se implemente pago en línea, los datos personales de clientes (nombre, contacto) deben resguardarse. Si en el futuro se agrega pago con tarjeta, el sistema deberá cumplir estándares de seguridad (p. ej., PCI DSS) para el manejo de datos financieros.

Detalles adicionales

- Tipo de usuario final: Clientes del cine (que reservan o compran entradas) y empleados de la taquilla/administración del cine (que utilizan el sistema para gestionar funciones, ventas y reservas telefónicas).
- Entorno de uso: Aplicación accesible vía web; los clientes pueden usarla desde navegadores en sus computadoras o smartphones. En el cine, los empleados usan PCs o terminales en la boletería, todos conectados a la misma base de datos central para sincronización en tiempo real.
- Tecnologías sugeridas: Desarrollo web con un componente frontend interactivo (por ejemplo, JavaScript/HTML5 para mostrar la selección de asientos) y un backend (Node.js, Java Spring, etc.) con base de datos

transaccional (SQL) que maneje la lógica de reservas. Posible integración futura con una pasarela de pago en línea (no obligatoria en la primera versión).

- Restricciones de tiempo: Se puede abordar en un proyecto de un cuatrimestre. Plan de iteraciones: primeramente, gestionar funciones y reservas básicas (sin interfaz gráfica de asientos), luego implementar la visualización gráfica de salas y control de concurrencia en reservas, seguido de módulo de cancelaciones y eventualmente notificaciones.
- Consideraciones: Se asumirá un cine pequeño (ej., 3 salas de ~100 asientos cada una). No se incluirán características avanzadas como venta de comida o programas de lealtad para mantener el alcance enfocado en la reserva de entradas. La verificación de boletos en la entrada (código QR u otros) se podría mencionar como posible extensión, pero no se implementará en el alcance inicial.

PROYECTO 6
Sistema de gestión de cultivos para una granja

Descripción

La granja “AgroSan” se dedica al cultivo de hortalizas y frutas en varias parcelas. Actualmente, los agricultores llevan un cuaderno de campo para anotar fechas de siembra, riego, fertilización y cosecha, y usan su memoria o llamadas para coordinar tareas diarias. Esto produce desorganización: a veces se olvidan labores o no se sabe con precisión el rendimiento de cada cosecha. Se necesita un sistema de gestión de cultivos que permita planificar y hacer seguimiento de las actividades agrícolas (siembra, riego, cosecha), registrando también las producciones obtenidas, para optimizar la operación de la granja.

Requerimientos funcionales

- RF1: Registrar cultivos plantados con su información básica (tipo de cultivo, fecha de siembra, parcela o terreno asignado, cantidad o área sembrada). Criterio de aceptación: Al ingresar un nuevo cultivo, el sistema guarda el registro con todos sus datos y lo incluye en la lista de cultivos activos, identificando la parcela y fechas relevantes (siembra y estimada de cosecha).
- RF2: Calendarizar y gestionar actividades agrícolas (riego, fertilización, aplicación de pesticidas) asociadas a cada cultivo. Criterio de aceptación: Para un cultivo registrado, el usuario puede programar una actividad (ej., “Riego cada lunes” o “Aplicar fertilizante tal fecha”); el sistema registra esta programación y, llegado el día, muestra un recordatorio o marca la actividad como pendiente para esa parcela/cultivo.
- RF3: Generar alertas o recordatorios para hitos importantes, como fechas estimadas de cosecha. Criterio de aceptación: Dado un cultivo con fecha estimada de cosecha, una semana antes de dicha fecha el sistema envía una notificación al agricultor (o la muestra en el panel) indicando que se aproxima la cosecha, incluyendo el cultivo y parcela involucrada, para preparar la recolección.
- RF4: Registrar la cosecha y el rendimiento obtenido. Criterio de aceptación: Cuando se realiza la cosecha de un cultivo, el usuario ingresa la cantidad obtenida (por ejemplo, kilogramos de producto). El sistema marca el cultivo como “cosechado”, almacena la producción real obtenida y permite consultarla en reportes históricos, comparándola con lo esperado si se ingresó un estimado.
- RF5: Llevar control de la distribución o venta básica de la cosecha. Criterio de aceptación: Tras registrar una cosecha, el sistema permite anotar si la

producción fue vendida (cantidad, fecha, comprador o destino). Al guardar estos datos, se genera un registro de salida (venta o uso interno) de la producción, de forma que el agricultor puede ver cuánto producto se ha vendido y cuánto queda en inventario o almacenamiento, si aplica.

Requerimientos no funcionales

- Usabilidad: La aplicación debe ser sencilla para usuarios con conocimientos básicos de informática (agricultores). Idealmente ofrecer una interfaz simple, con íconos o menús claros para registrar actividades en el campo, incluso mediante una tableta o smartphone, considerando que se usará en exteriores.
- Operatividad sin conexión: Dado que la granja puede estar en una zona con conectividad limitada, el sistema (o app móvil) debería poder registrar datos offline y sincronizar cuando haya conexión, o al menos funcionar en una red local dentro de la granja sin internet.
- Fiabilidad: Debe prevenir pérdida de datos de cultivos y cosechas; por ejemplo, realizando respaldos frecuentes de la información (diariamente o después de cada carga de datos significativa) para conservar el historial incluso si ocurre una falla o apagón.
- Flexibilidad: El sistema debe permitir agregar nuevos tipos de cultivo o modificar calendarios de actividades sin necesidad de reprogramación compleja (idealmente mediante opciones de configuración en la interfaz), de modo que la herramienta siga siendo útil si la granja diversifica sus cultivos o cambia prácticas.
- Seguridad: Aunque es un entorno pequeño, se debe proteger el acceso con usuario/contraseña para evitar que personas no autorizadas alteren los registros. Los datos históricos de producción son sensibles para el negocio y deben mantenerse confidenciales dentro de la empresa.

Detalles adicionales

- Tipo de usuario final: Propietario o administrador de la granja, y eventualmente supervisores de campo que reportan datos (p. ej., podrían usar la aplicación para marcar actividades realizadas).
- Entorno de uso: Oficina de la granja (PC de escritorio con la base de datos principal) y campo abierto (posible uso en dispositivo móvil para anotar eventos in situ). Puede funcionar como aplicación de escritorio autónoma o web local en la red de la granja.
- Tecnologías sugeridas: Una aplicación de escritorio (por ejemplo, Java con SQLite) por su sencillez y posibilidad de uso offline, o una app móvil nativa sencilla para Android con sincronización diferida. Alternativamente, una

aplicación web progresiva (PWA) que funcione offline para dispositivos móviles y sincronice cuando detecte conexión.

- Restricciones de tiempo: Proyecto realizable en un semestre (~4-5 meses). Se sugiere priorizar en etapas: primero registro de cultivos y cosechas, luego módulo de actividades programadas (riego, fertilización), y finalmente registro de ventas/distribución.
- Consideraciones: Se asumirá una granja pequeña o mediana (por ejemplo, 10 parcelas y 5 tipos principales de cultivo) para mantener el alcance manejable. No se integrarán sensores IoT ni sistemas externos; el enfoque está en registro manual y seguimiento básico. Las métricas avanzadas (ej. eficiencia por cultivo, costos) podrían considerarse extensiones futuras, pero no forman parte de los requerimientos iniciales.

PROYECTO 7
Sistema de reservas para agencia de turismo

Descripción

“Adventure Tours” es una agencia de turismo local que organiza excursiones y tours en la región. Actualmente gestiona las reservas de sus tours vía correo electrónico y planillas manuales, lo que ocasiona problemas de sobreventa de cupos, dificultad para hacer seguimiento a pagos de clientes y desorden en la logística de cada tour. Se propone un sistema de reservas para la agencia de turismo que centralice la oferta de tours, permita registrar las reservas de clientes con sus pagos asociados, y facilite la administración de cupos disponibles por actividad.

Requerimientos funcionales

- RF1: Registrar paquetes o tours con sus detalles (destino/actividad, descripción, fecha(s) de salida, duración, ubicación, precio por persona, cupo máximo de participantes). Criterio de aceptación: Al crear un nuevo tour en el sistema con todos sus datos, este queda disponible para reservas, mostrando el nombre del tour, fecha de salida programada y número de plazas disponibles inicialmente igual al cupo máximo definido.
- RF2: Permitir reservar un cupo para clientes en un tour determinado. Criterio de aceptación: Cuando un agente ingresa una reserva para X personas en un tour específico, el sistema descuenta ese número del cupo disponible del tour y guarda la información del cliente principal (nombre, contacto) y el número de participantes reservados. Tras la operación, el tour muestra su disponibilidad actualizada.
- RF3: Controlar que no se exceda el cupo máximo en las reservas. Criterio de aceptación: Si se intenta reservar más plazas de las disponibles para un tour, el sistema no lo permite y emite un mensaje de error indicando que el cupo es insuficiente, manteniendo inalterados los datos de disponibilidad.
- RF4: Facilitar la cancelación o modificación de reservas. Criterio de aceptación: Dada una reserva existente, el usuario (agente) puede cancelarla (liberando las plazas reservadas y devolviéndolas al cupo disponible del tour) o modificarla (por ejemplo, reducir la cantidad de personas o cambiar al cliente a otro tour si es factible). El sistema actualiza los datos en consecuencia y registra la modificación realizada.
- RF5: Registrar el estado de pago de cada reserva. Criterio de aceptación: Al ingresar una reserva, el sistema permite marcar su estado de pago (pagado, pago parcial con saldo, o pendiente). Por ejemplo, una reserva puede registrarse con un depósito inicial pagado; el sistema reflejará “pagado

parcialmente” y permitirá actualizar a “pagado completo” cuando el cliente abone el resto. Esto garantiza un seguimiento claro de qué reservas están pagadas y cuáles no.

Requerimientos no funcionales

- Usabilidad: Debe poseer una interfaz clara para los agentes de viajes, que les permita buscar rápidamente un tour por nombre o fecha y realizar la reserva en pocos pasos mientras atienden al cliente. Formularios y procesos sencillos reducirán errores en la toma de datos.
- Eficiencia: Las consultas de disponibilidad de tours y actualizaciones de reservas deben ser rápidas (< 2 segundos), asegurando que el agente pueda confirmar al cliente la reserva casi inmediatamente. El sistema debe soportar entradas concurrentes de dos o más agentes sin inconsistencias (evitando doble reserva del mismo cupo).
- Confiabilidad: Es crucial mantener la consistencia de los datos de reservas. El sistema debe protegerse contra caídas durante la operación (por ejemplo, guardando transacciones de reserva de forma atómica, de modo que no queden a medio camino). Deben implementarse backups regulares de la base de datos de reservas para recuperar información en caso de fallo.
- Seguridad: Los datos de los clientes (nombres, contactos, estado de pagos) deben estar protegidos con controles de acceso. Solo el personal de la agencia con credenciales puede acceder al sistema. Además, si en el futuro se maneja información de tarjetas de crédito para pagos en línea, dicha información deberá cifrarse y manejarse según estándares de seguridad.
- Escalabilidad: El diseño debe admitir que la agencia amplíe su oferta (más tours, múltiples fechas, incluso sucursales adicionales) sin degradación notable del rendimiento. También se podría prever la futura incorporación de un portal web para que clientes reserven directamente, con el mínimo esfuerzo de integración, usando la misma base de datos.

Detalles adicionales

- Tipo de usuario final: Empleados de la agencia de turismo (agentes de viajes que toman reservas y administran la oferta de tours, y posiblemente un gerente que supervisa). No hay portal abierto a clientes externos en esta fase, aunque podría considerarse como extensión futura.
- Entorno de uso: Oficinas de la agencia con computadoras de escritorio o portátiles; el sistema puede alojarse en la nube o en un servidor local, pero debería ser accesible vía Internet segura para posibles consultas fuera de la oficina (ej., un agente que trabaja remoto o en una feria turística).

- Tecnologías sugeridas: Aplicación web centralizada con base de datos relacional (MySQL, PostgreSQL) para las reservas y tours. Un framework web (por ejemplo, Django, Laravel) facilitaría la implementación rápida de formularios y listados. Se pueden usar bibliotecas para exportar confirmaciones en PDF o enviar emails de confirmación a los clientes (vía SMTP).
- Restricciones de tiempo: Proyecto de 4-5 meses. Iteraciones sugeridas: primero módulo de creación de tours y control de cupos, luego módulo de registro de reservas, seguido por cancelaciones/modificaciones, y finalmente el control de pagos y notificaciones. Pruebas con escenarios simulados (varios agentes reservando al mismo tiempo) para asegurar consistencia.
- Consideraciones: Se acotará el alcance a tours sencillos de uno o pocos días sin logística compleja (no se considerará gestión detallada de hoteles, vuelos u otros servicios adicionales). La agencia modelo maneja ~20 tours distintos al mes con grupos de hasta 15 personas, lo cual sirve de referencia para dimensionar los casos de uso y pruebas.

PROYECTO 8
Sistema de control de pedidos en un taller de manufactura

Descripción

“Mecafab” es un pequeño taller de manufactura que fabrica piezas y productos a pedido para clientes locales (por ejemplo, partes metálicas o equipos personalizados). Actualmente, los pedidos de clientes se registran en formularios de papel y un pizarrón en el taller muestra el estado de cada trabajo. Este método manual causa desorden: se han extraviado pedidos, algunos trabajos se retrasan sin que el gerente se entere a tiempo, y es difícil obtener un panorama general de la carga de trabajo. Se propone un sistema de control de pedidos que digitalice el registro y seguimiento de cada orden de trabajo, desde que el cliente la solicita hasta que se entrega, mejorando la planificación en el taller.

Requerimientos funcionales

- RF1: Registrar nuevos pedidos de clientes con sus especificaciones (descripción del producto a fabricar o servicio a realizar, datos del cliente, fecha de solicitud y fecha comprometida de entrega). Criterio de aceptación: Al ingresar un pedido con todos los datos requeridos, el sistema genera un identificador único de pedido y lo almacena; el pedido aparece en la lista de “pendientes” con su estado inicial (por ejemplo, “Registrado” o “En espera de inicio”).
- RF2: Asignar el pedido a un proceso de producción o a un encargado. Criterio de aceptación: Desde la lista de pedidos, el usuario (jefe de taller) puede asignar el pedido a un operario o equipo específico para su fabricación; el sistema registra el responsable asignado y cambia el estado del pedido a “En producción”, registrando la fecha de inicio de trabajo.
- RF3: Actualizar el estado de cada pedido a medida que avanza en el taller. Criterio de aceptación: Los usuarios pueden marcar un pedido con estados intermedios relevantes (por ejemplo, “En proceso”, “En espera de material”, “En acabado”, “Completado”). Por ejemplo, al terminar la fabricación, se marca como “Completado” con la fecha de finalización; el sistema guarda este estado final para el historial.
- RF4: Generar alertas de próximos vencimientos o retrasos. Criterio de aceptación: Si un pedido no se ha completado y faltan 2 días o menos para su fecha comprometida de entrega (o si ya la sobrepasó), el sistema destaca ese pedido (p.ej., con color de alerta) en la interfaz y puede enviar una notificación al jefe de taller indicando que la fecha límite está cercana o ha sido excedida.
- RF5: Registrar la entrega final del pedido al cliente y preparar documentación

básica. Criterio de aceptación: Una vez completado, el sistema permite marcar el pedido como “Entregado” cuando el producto ha sido dado al cliente. Al hacerlo, genera un resumen del pedido (detalles del producto, fecha de entrega real, nombre del cliente) que puede imprimirse como comprobante o utilizar si se requiere emitir factura externa. El pedido se archiva como finalizado en el sistema.

Requerimientos no funcionales

- Usabilidad: Interfaz accesible para personal no necesariamente experto en informática (operarios o jefes de taller). Debe tener pantallas simples: una lista de pedidos con botones para cambiar estado, formularios directos para ingresar un nuevo pedido, etc., minimizando la posibilidad de error.
- Eficiencia: Dado el volumen moderado (un taller pequeño podría manejar decenas de pedidos abiertos simultáneamente), el sistema debe mostrar la lista y permitir actualizaciones sin demoras perceptibles. Consultar o filtrar pedidos (por estado, por cliente) debe ser posible en tiempo real para que el jefe tome decisiones rápidas.
- Confiabilidad: La aplicación debe mantener actualizados los datos de estado sin inconsistencias. Debe tolerar fallos aislados (por ejemplo, si se cierra inesperadamente, los pedidos no deben perderse ni corromperse). Respaldos periódicos (por ejemplo, cada noche) asegurarán que se pueda restaurar la información tras una falla grave.
- Seguridad: Los datos de pedidos y clientes son internos a la empresa; el sistema requerirá autenticación para acceder. Podría implementarse control de roles, de modo que solo supervisores puedan borrar o marcar entregas finales, mientras operarios solo actualizan estados intermedios. Esto previene modificaciones no autorizadas o accidentales en datos críticos.
- Mantenibilidad: Deberá ser fácil de actualizar. Por ejemplo, agregar nuevos estados de pedido o adaptar el flujo si cambian los procesos del taller no debería requerir reescribir desde cero. Un diseño modular y una buena documentación del código ayudarán a futuros desarrolladores a modificar o ampliar el sistema.

Detalles adicionales

- Tipo de usuario final: Jefe de taller o encargado de producción, operarios clave del taller que reportan estados, y posiblemente personal administrativo que consulta el avance de pedidos. Es un sistema de uso interno.
- Entorno de uso: Computadora de escritorio en la oficina del taller (para registro de pedidos y seguimiento por el jefe) y potencialmente una terminal o

tableta en el área de producción para que operarios actualicen estados. Puede funcionar en una red local sin necesidad de internet (intranet del taller).

- Tecnologías sugeridas: Aplicación de escritorio multiusuario en red local (por ejemplo, .NET/C# con base de datos SQL Server Express, o Java con base de datos H2/Derby) o aplicación web interna. Se puede comenzar con una base de datos ligera (SQLite) para prototipo y migrar a una más robusta si el volumen crece.
- Restricciones de tiempo: Proyecto para un curso de 4-5 meses. Iteraciones recomendadas: primero módulo de registro y listado de pedidos, luego actualización de estados, después alertas de vencimiento, y finalmente módulo de cierre/entrega y roles de usuario. Probar con datos simulados (ej., 50 pedidos) para verificar comportamientos.
- Consideraciones: El alcance no incluye gestión de inventarios de materias primas ni contabilidad; se limita al flujo de pedidos/productos en taller. Se asume un taller pequeño (3-5 empleados manejando ~50 pedidos al mes) para dimensionar los requisitos. La integración con sistemas de facturación se podría mencionar, pero no se implementará (solo se prevé un comprobante básico de entrega).

PROYECTO 9
Sistema de gestión de presupuesto personal

Descripción

Juan Pérez, un profesional independiente, desea organizar mejor sus finanzas personales. Actualmente lleva sus gastos e ingresos anotados en papel o en notas dispersas, lo que dificulta tener una visión clara de su situación financiera mensual, planificar ahorros o identificar en qué se está gastando más. Por ello, identifica la necesidad de un sistema de gestión de presupuesto personal que le permita registrar ingresos y egresos, clasificarlos por categoría, y obtener reportes simples que le ayuden a tomar decisiones financieras informadas.

Requerimientos funcionales

- RF1: Registrar transacciones de ingreso con detalles (fecha, monto, categoría, descripción). Criterio de aceptación: Al ingresar un nuevo ingreso con todos sus datos, el sistema lo almacena, actualiza el saldo total acumulado, y el ingreso aparece listado en el historial con la categoría asignada.
- RF2: Registrar transacciones de gasto con detalles similares. Criterio de aceptación: Cuando el usuario registra un gasto indicando fecha, monto y categoría (por ejemplo, “Alimentación”, “Transporte”), el sistema guarda la transacción y recalcula el balance. El gasto queda visible en el historial y su monto se descuenta del total o presupuesto correspondiente.
- RF3: Permitir la gestión de categorías de gasto/ingreso. Criterio de aceptación: El usuario puede crear, editar o eliminar categorías personalizadas (p. ej., “Alquiler”, “Entretenimiento”). Al agregar una nueva categoría, esta aparece como opción al registrar transacciones y puede ser utilizada inmediatamente, reflejándose también en los reportes por categoría.
- RF4: Generar reportes financieros periódicos (mensuales, trimestrales) y por categoría. Criterio de aceptación: Al solicitar un resumen mensual, el sistema muestra el total de ingresos, el total de gastos y el saldo neto del mes seleccionado, además de un desglose por categoría (p. ej., cuánto se gastó en cada categoría ese mes), presentando los datos en una tabla o gráfica simple.
- RF5: Manejar presupuestos o alertas de gasto excesivo. Criterio de aceptación: El usuario puede fijar un presupuesto máximo para una categoría o para gastos mensuales totales. Si al registrar un nuevo gasto el acumulado del mes en esa categoría supera el presupuesto definido, el sistema emite una alerta indicando que se ha excedido el monto previsto.

Requerimientos no funcionales

- Usabilidad: La aplicación debe ser muy intuitiva, apta para un usuario individual sin conocimientos contables profundos. Por ejemplo, flujos simplificados para agregar un gasto con pocos clics, y visualizaciones claras (gráficos básicos) para entender la distribución de gastos por categoría. Una interfaz atractiva y sencilla motivará su uso constante.
- Accesibilidad: Sería beneficioso que la aplicación esté disponible en múltiples plataformas (por ejemplo, versión móvil y versión web) para que el usuario pueda registrar gastos en el momento (desde el teléfono) y consultar o analizar su presupuesto en detalle desde una computadora. Los datos deben sincronizarse de forma segura entre dispositivos si se usa más de uno.
- Seguridad y Privacidad: La información financiera personal es sensible; si la aplicación requiere inicio de sesión, debe proteger la cuenta con autenticación robusta (contraseña segura, potencial segundo factor o biometría en móvil). Los datos almacenados (sobre todo si están en la nube) deben cifrarse para prevenir accesos no autorizados. La política de privacidad debe ser clara respecto a que los datos no se compartirán sin consentimiento.
- Confiabilidad: Debe evitarse la pérdida de datos. La app podría incluir copias de seguridad automáticas (por ejemplo, sincronización en la nube o exportación periódica a un archivo) para asegurar que, ante una falla del dispositivo, la información financiera no se pierda. Además, las operaciones de registro de transacciones deben ser atómicas (completamente guardadas o no guardadas) para que no queden registros inconsistentes si la aplicación se cierra repentinamente.
- Rendimiento: Aunque el volumen de datos es relativamente bajo (transacciones diarias), la aplicación debe mantener un rendimiento fluido. Las consultas de reportes, incluso acumulando datos de ... Rendimiento: Las consultas de reportes (incluso con varios años de datos) deberían generarse en segundos, y la interfaz debe responder ágilmente a la interacción del usuario para no frustrar su uso continuo.

Detalles adicionales

- Tipo de usuario final: Persona individual (ej. Juan Pérez u otros profesionales) que administra sus finanzas; podría extenderse a una familia pequeña que comparte el presupuesto doméstico.
- Entorno de uso: Dispositivo personal como un smartphone (Android/iOS) para anotar gastos sobre la marcha, y/o una aplicación web o de escritorio para análisis más detallado en casa. Es deseable disponibilidad multiplataforma con sincronización de datos.

- Tecnologías sugeridas: Aplicación móvil nativa (p. ej., Android Studio para Android) con base de datos local y sincronización en la nube (Firebase u otro servicio) para backup. Alternativamente, una aplicación web PWA que funcione offline con almacenamiento local (IndexedDB) y se sincronice cuando haya conexión.
- Consideraciones: No se implementarán funciones contables avanzadas (cálculo de impuestos, inversión, etc.); el alcance se limita al seguimiento simple de ingresos y gastos clasificados. El diseño debe motivar al usuario a registrar sus gastos diariamente, ya que la constancia es clave para obtener valor de la herramienta.

PROYECTO 10
Sistema de gestión de gimnasio y suscripciones

Descripción

El “Gimnasio VidaFit” es un centro deportivo que desea modernizar el control de sus socios y actividades. Actualmente utilizan tarjetas físicas y hojas de cálculo para llevar registro de pagos y accesos, lo que provoca confusiones con membresías vencidas y dificulta coordinar las clases grupales. Se propone un sistema de gestión de gimnasio que administre la información de socios, el estado de sus suscripciones, controle su ingreso diario y gestione la inscripción a clases, mejorando la operación y servicio al cliente.

Requerimientos funcionales

- RF1: Registrar nuevos socios con sus datos personales y tipo de suscripción. Criterio de aceptación: Al ingresar un nuevo miembro con nombre, contacto y plan (ej. mensual, trimestral), el sistema guarda su perfil y establece su estado “Activo” con fecha de expiración según el plan. El socio aparece en la lista de miembros con su vigencia correspondiente.
- RF2: Gestionar pagos y vigencia de las membresías. Criterio de aceptación: Cuando un socio paga su cuota o renovación, el personal registra el pago; el sistema actualiza la fecha de vencimiento de la membresía. Si un socio supera su fecha de expiración sin pago, el sistema cambia automáticamente su estado a “Inactivo” (impidiendo su ingreso) hasta que regularice el pago.
- RF3: Controlar el acceso diario de socios mediante registro de entradas. Criterio de aceptación: Al llegar un socio al gimnasio, el recepcionista puede buscarlo por nombre o ID (o escanear su carnet) y marcar su ingreso. El sistema verifica que esté “Activo” y registra la hora de entrada. Si el socio está “Inactivo”, muestra una alerta indicando que la membresía está vencida.
- RF4: Administrar la inscripción a clases o actividades grupales. Criterio de aceptación: El personal puede crear sesiones de clases (ej. yoga lunes 6pm con cupo 15). Un socio activo puede ser inscrito en una clase; el sistema reduce el cupo disponible. Si el cupo está lleno, no permite más inscripciones. La lista de inscritos es visible para el instructor.
- RF5: Enviar notificaciones o recordatorios a los socios. Criterio de aceptación: El sistema genera alertas cuando un socio está próximo a vencer su membresía (ej. 5 días antes) para que el personal lo notifique o enviar un email automático. Igualmente, si un socio inscrito a una clase tiene la clase próxima, se puede enviar un recordatorio con la hora y sala (opcional en el alcance inicial).

Requerimientos no funcionales

- Usabilidad: La interfaz de recepción debe permitir identificar a un socio rápidamente (búsqueda instantánea por nombre/ID). Debe ser compatible con uso de lectores de código de barras o QR en carnés, aunque también funcionar con búsqueda manual. La curva de aprendizaje para el personal debe ser mínima.
- Disponibilidad: El sistema debe estar disponible durante todo el horario de apertura del gimnasio (por ejemplo, 6am–10pm todos los días). Cualquier mantenimiento debe programarse fuera de horas de servicio para no impedir la verificación de accesos.
- Rendimiento: La búsqueda de un socio o verificación de entrada debe ser prácticamente instantánea (<1 segundo) dado que puede haber fila en recepción. El sistema debe soportar una base de datos de varios miles de socios y registros de asistencia sin lentitud notable.
- Seguridad: Los datos personales de los socios (contacto, historial de pagos, asistencia) deben protegerse mediante autenticación y control de acceso. Por ejemplo, solo administradores pueden ver/editar información financiera o expirar manualmente una membresía, mientras que recepcionistas solo registran entradas y pagos.
- Escalabilidad: Si el gimnasio crece o abre sucursales, el sistema debería poder adaptarse a múltiples locales y más usuarios concurrentes. Por diseño, conviene separar la lógica de negocio de la interfaz para facilitar agregar en el futuro una app móvil para socios (consulta de estado, inscripciones) con la misma base de datos.

Detalles adicionales

- Tipo de usuario final: Empleados del gimnasio: recepcionistas (registro de accesos y pagos), administradores/gerentes (gestión de membresías, informes), instructores (consulta de lista de inscritos a clases). Los socios no usan directamente el sistema en esta versión.
- Entorno de uso: Mostrador de recepción con PC conectado a la base de datos central; posibilidad de varios puestos (recepción, administración) en red local o en la nube sincronizados. Integración con hardware (lector de tarjetas código de barras/RFID) podría usarse pero no es indispensable inicialmente.
- Tecnologías sugeridas: Arquitectura cliente-servidor. Por ejemplo, backend en Python (Django) o Node.js con base de datos PostgreSQL, y frontend web accesible desde los PCs de recepción. Alternativamente, una aplicación de escritorio en .NET con base de datos compartida en red local. Envío de emails automatizados usando un servicio SMTP o API.

- Consideraciones: No se integrará en esta versión el procesamiento de pagos en línea (se asume que las cuotas se pagan en persona y luego se registran). Tampoco se gestionarán aspectos como rutinas de ejercicio ni control de aparatos; el alcance se limita a membresías, accesos y clases grupales básicas.

PROYECTO 11
Sistema de pedidos y reservas para restaurante

Descripción

El restaurante “Delicias Gourmet” desea mejorar la organización de sus mesas y pedidos. Actualmente manejan las reservas de mesas en un cuaderno y los meseros anotan pedidos en papel, lo que provoca errores (pedidos olvidados, cuentas incorrectas) y dificulta conocer la disponibilidad de mesas en todo momento. Se propone un sistema integral para el restaurante que permita registrar reservas de mesas, tomar pedidos de los comensales y hacer seguimiento hasta la facturación, agilizando el servicio y reduciendo errores.

Requerimientos funcionales

- RF1: Gestionar reservas de mesas por fecha y hora. Criterio de aceptación: El sistema permite al encargado registrar una reserva con nombre del cliente, fecha/hora y número de personas, asignando una mesa. Al guardar, esa mesa queda marcada como reservada en ese horario y aparece en la lista de reservas del día. Si se intenta reservar una mesa ya ocupada/reservada para esa hora, el sistema no lo permitirá.
- RF2: Registrar la llegada de clientes y asignación de mesas, incluyendo reservas. Criterio de aceptación: Cuando llega un cliente con reserva, el mesero puede marcar que la reserva se hace efectiva (ocupando la mesa). Si llegan clientes sin reserva, el mesero verifica una mesa libre y crea un registro de “mesa ocupada” indicando hora de inicio y número de comensales.
- RF3: Tomar pedidos de comida y bebida por mesa. Criterio de aceptación: El mesero selecciona en la interfaz la mesa ocupada y añade los ítems del menú al pedido de esa mesa. Al confirmar, el pedido queda registrado como “En preparación” y aparece en la lista de pedidos de cocina con la hora y mesa correspondientes.
- RF4: Actualizar el estado de los pedidos de mesa (en preparación, listo, servido). Criterio de aceptación: El cocinero o encargado de cocina puede marcar un pedido como “Listo” cuando esté preparado; el mesero ve esta actualización y, tras servirlo, marca el pedido como “Servido”. El sistema registra las horas de estos cambios para monitorear tiempos de servicio.
- RF5: Generar la cuenta/factura de una mesa. Criterio de aceptación: Al solicitar la cuenta, el sistema compila todos los ítems consumidos por la mesa con sus precios y calcula el total. Esta información puede imprimirse o mostrarse para cobro. Una vez registrada la salida/pago, la mesa se libera (vuelve a estado disponible) y el sistema guarda el historial de la venta.

Requerimientos no funcionales

- Usabilidad: La aplicación debe ser ágil de usar en un entorno de alta rotación. Interfaz optimizada para pantallas táctiles (tablets) con botones grandes y flujos rápidos para tomar pedidos y actualizar estados, de modo que el mesero no navegue por múltiples pantallas. La capacitación de los meseros en el sistema debe ser casi inmediata.
- Rendimiento: Las operaciones deben reflejarse inmediatamente en todos los terminales. Por ejemplo, cuando cocina marca un pedido como listo, el mesero debe verlo al instante. El sistema debe funcionar fluidamente en horas pico (docenas de pedidos concurrentes) sin retrasos perceptibles.
- Confiabilidad: El sistema no debe fallar durante el servicio. Se recomienda una instalación local (servidor en el restaurante) para no depender de Internet. Debe haber mecanismos de recuperación: si un dispositivo se desconecta, los pedidos deben quedar guardados en el servidor y sincronizarse al reconectar. Respaldos diarios de la base de datos asegurarán conservar las ventas históricas.
- Seguridad: Solo personal autorizado puede usar el sistema; cada mesero/cajero tendrá sus credenciales. Ciertas acciones (ej. anular una cuenta o dar descuentos) podrían requerir permisos de gerente para evitar abusos. Además, la información de ventas y estadísticas debe estar protegida para uso interno únicamente.
- Escalabilidad y compatibilidad: Debe soportar múltiples dispositivos concurrentes (varios meseros con tablet, una pantalla en cocina, un terminal de caja) sin conflictos. Ser compatible con distintos dispositivos (tabletas Android, iPad, PC) para flexibilidad. A futuro, si el restaurante abre otra sucursal, el sistema podría adaptarse a manejar múltiples locales separados en la misma plataforma.

Detalles adicionales

- Tipo de usuario final: Personal del restaurante: meseros (gestionan mesas y pedidos), personal de cocina (visualizan y actualizan pedidos), cajero/administrador (emite cuentas, supervisa reservas y ventas). Los clientes no interactúan directamente con el sistema.
- Entorno de uso: Dentro del restaurante con varios dispositivos conectados en red local (Wi-Fi). Meseros con tablets o smartphones empresariales; cocina con una pantalla o impresora para ver pedidos; caja con un PC o terminal punto de venta. La red local garantiza baja latencia; puede haber conexión a Internet para respaldo pero no debe ser necesaria para la operación diaria.
- Tecnologías sugeridas: Sistema cliente-servidor en red local. Por ejemplo, un

servidor central (PC) ejecutando una aplicación web (Node.js, Java Spring o similar con base de datos MySQL) al que se conectan clientes vía navegador web o app. Opcional: uso de impresoras de cocina y lectores de código QR para cuentas, integrados mediante módulos específicos.

- Consideraciones: No se incluirá inicialmente la gestión de inventario de ingredientes ni un módulo de compras; el proyecto se enfoca en la operación de piso (mesas, pedidos y cobros). Se asume un restaurante mediano (~20 mesas, 5 meseros por turno, 2 cocineros) para dimensionar el sistema. Se podrían obtener estadísticas básicas de ventas por día/mes a partir de los datos registrados, aunque ese no sea el foco principal del sistema.

PROYECTO 12
Sistema de gestión inmobiliaria (propiedades y clientes)

Descripción

La agencia inmobiliaria “Hogar Seguro” maneja inmuebles en venta y alquiler, y desea un mejor control de sus listados y clientes interesados. Actualmente utilizan carpetas compartidas y notas para hacer seguimiento, lo que dificulta saber qué propiedades están disponibles, cuándo se mostrará cada una y a quién. Se plantea un sistema de gestión inmobiliaria que centralice el catálogo de propiedades, los datos de clientes interesados y las citas de visita, facilitando el seguimiento de cada oportunidad de negocio.

Requerimientos funcionales

- RF1: Registrar propiedades disponibles con sus datos (dirección, tipo: casa/depto, características principales, precio, estado disponible/vendido/alquilado). Criterio de aceptación: Al ingresar una nueva propiedad con sus campos, esta queda almacenada y aparece en el catálogo con estado “Disponible”. Si se registran múltiples unidades (ej. departamentos), cada una tiene su ficha individual.
- RF2: Permitir la búsqueda y filtrado de propiedades según criterios. Criterio de aceptación: El sistema ofrece filtros por ubicación, rango de precio, tipo de propiedad, número de habitaciones, etc. Por ejemplo, si un agente busca "departamento, 2 dormitorios, $100k–$150k", la aplicación muestra la lista de propiedades que cumplen esos criterios, con información resumida y disponibilidad.
- RF3: Gestionar la información de clientes interesados y sus requerimientos. Criterio de aceptación: Un agente puede registrar un nuevo cliente potencial con sus datos de contacto y preferencias (ej. "busca casa 3 habitaciones con jardín, presupuesto hasta $200k"). Estos datos quedan guardados y se pueden vincular a propiedades que le interesaron o se le mostraron, para seguimiento.
- RF4: Programar y registrar visitas para mostrar propiedades. Criterio de aceptación: Dado un cliente y una propiedad de interés, el agente puede agendar una visita indicando fecha y hora; el sistema la registra en la agenda. Si la visita genera conflicto (mismo agente con dos citas a la vez, o dos clientes para la misma propiedad a la misma hora), el sistema no lo permitirá. Tras realizar la visita, el agente puede marcarla como realizada y agregar notas (ej. nivel de interés del cliente).
- RF5: Actualizar el estado de las propiedades y registrar operaciones

finalizadas. Criterio de aceptación: Si una propiedad se vende o alquila, el agente la marca en el sistema como “Vendido” o “Alquilado” con la fecha correspondiente. El sistema la retira de la lista de disponibles y guarda el registro de la transacción (por ejemplo, comprador/arrendatario y precio final) para consultas internas futuras.

Requerimientos no funcionales

- Usabilidad: El sistema debe ofrecer a los agentes inmobiliarios una interfaz clara para manejar gran cantidad de información. Por ejemplo, vistas separadas para Propiedades y Clientes, con búsquedas rápidas y formularios fáciles de llenar. Debería permitir visualizar el detalle completo de una propiedad (incluso fotos, si se almacenan) de forma ordenada, y registrar una cita en pocos pasos.
- Rendimiento: Las búsquedas en el catálogo de propiedades deben ser rápidas (incluso con cientos de listados, resultados en < 2 segundos). La agenda de visitas debe cargar ágilmente. Dado que probablemente habrá pocos usuarios concurrentes (2–5 agentes), el sistema no debería experimentar problemas de latencia.
- Confiabilidad: La información de propiedades y clientes es valiosa; se deben realizar respaldos frecuentes para no perder datos de contacto o historiales. La integridad referencial debe cuidarse: por ejemplo, si se elimina una propiedad del catálogo, asegurarse de conservar o manejar las citas asociadas para no perder registro de lo actuado.
- Seguridad: Se manejarán datos personales de clientes y detalles confidenciales (como precios de cierre). El acceso al sistema debe requerir autenticación, y podría haber roles (agentes comunes vs gerente) donde quizás solo el gerente puede eliminar propiedades o ver el registro completo de transacciones. Toda comunicación remota debe ir cifrada si el sistema se usa fuera de la oficina (HTTPS).
- Escalabilidad: Si la agencia crece (más propiedades o más agentes), el sistema debería soportar la carga adicional aumentando la capacidad de la base de datos y permitiendo más concurrencia. También debería prever posible integración futura con un portal web público donde los clientes vean propiedades (extrayendo datos del mismo sistema para no duplicar información).

Detalles adicionales

- Tipo de usuario final: Agentes inmobiliarios y administradores de la agencia (usuarios internos). En esta fase no hay acceso para clientes externos; los

clientes interesados son gestionados por los agentes a través del sistema.

- Entorno de uso: Oficinas de la inmobiliaria (PCs de escritorio para los agentes) y también acceso remoto: los agentes podrían usar el sistema desde laptops o tablets mientras muestran propiedades (vía conexión internet móvil). Es deseable una aplicación web para acceder desde cualquier lugar con credenciales.
- Tecnologías sugeridas: Aplicación web tipo CRM ligero. Un stack LAMP (PHP/MySQL) o MERN (Mongo/Express/React/Node) sería adecuado. Uso de un framework web facilita funcionalidades estándar (autenticación, CRUD). Una base de datos SQL garantizará consistencia entre módulos (propiedades, clientes, citas).
- Consideraciones: No se incluyen funcionalidades de gestión contable (comisiones, pagos) ni generación de contratos legales; esas tareas se manejarán fuera del sistema. El alcance se limita a la gestión interna de listados, clientes y citas de visita. Se asume una agencia con ~100 propiedades activas y 5 agentes para escenarios de prueba.

PROYECTO 13
Sistema de gestión de biblioteca pública

Descripción

La Biblioteca Municipal “Lectura Viva” desea modernizar su sistema de préstamo de libros. Actualmente utiliza fichas en papel y un catálogo impreso, lo que dificulta llevar control preciso de los préstamos y devoluciones, y ofrecer búsqueda eficiente a los usuarios. Se propone un sistema de gestión bibliotecaria que administre el catálogo de libros, los socios inscritos y los préstamos de ejemplares, permitiendo conocer la disponibilidad de cada obra y agilizar la atención a los lectores.

Requerimientos funcionales

- RF1: Registrar libros en el catálogo con su información (título, autor, código/ISBN, temática, número de ejemplares disponibles). Criterio de aceptación: Al ingresar un nuevo libro con todos sus datos, este queda registrado con un identificador único; si se indica más de un ejemplar, el catálogo refleja el número de copias disponibles para préstamo.
- RF2: Gestionar usuarios de la biblioteca (socios inscritos) con sus datos personales. Criterio de aceptación: Cuando un bibliotecario registra un nuevo socio (nombre, identificación, contacto), el sistema almacena su perfil y le asigna un número de miembro. Ese usuario queda habilitado para realizar préstamos, evitando duplicados (no aceptar la misma identificación dos veces).
- RF3: Permitir el préstamo de libros, controlando disponibilidad y plazos. Criterio de aceptación: Al registrar un préstamo, el bibliotecario selecciona el socio y el libro deseado; el sistema verifica que haya ejemplares disponibles (no prestados) y crea el registro de préstamo asignando una fecha de devolución prevista (por ejemplo, 14 días después). El ejemplar prestado se descuenta de la disponibilidad.
- RF4: Registrar la devolución de libros. Criterio de aceptación: Cuando un socio devuelve un libro, el bibliotecario marca la devolución en el sistema; este libera el ejemplar (incrementa la disponibilidad en el catálogo) y registra la fecha de retorno. Si la devolución es posterior a la fecha prevista, el sistema marca el préstamo como “atrasado” en el historial (posibilitando cálculo de multa manual o aviso, aunque no lo calcule automáticamente).
- RF5: Permitir la búsqueda y consulta de libros en el catálogo por criterios (título, autor, tema). Criterio de aceptación: Si un usuario o bibliotecario realiza una búsqueda (por ejemplo, por autor o palabra clave), el sistema lista los libros que coinciden mostrando su información básica y disponibilidad

(ejemplares totales vs prestados). Los resultados deben aparecer ordenados de forma relevante y rápidamente (unos 2 segundos en búsquedas típicas).

Requerimientos no funcionales

- Usabilidad: El sistema debe ser fácil de usar para bibliotecarios. Por ejemplo, en préstamos/devoluciones, permitir búsqueda rápida por título o código de barras para identificar el libro, y por número de socio para identificar al usuario, minimizando tecleo. Una interfaz de catálogo público también debería ser intuitiva si se ofrece a visitantes.
- Rendimiento: La biblioteca puede tener miles de libros y cientos de usuarios, por lo que las búsquedas y operaciones deben ser eficientes. Consultar disponibilidad o registrar un préstamo debe ocurrir en pocos segundos. Debe soportar a varios bibliotecarios atendiendo simultáneamente sin lentitud (ej., 2–3 operando a la vez).
- Confiabilidad: Se debe garantizar la integridad de los registros de préstamos; nunca debe perderse la información de quién tiene qué libro. Respaldos diarios de la base de datos son esenciales. Además, el sistema debe manejar errores (caída de red, cierre inesperado) sin corromper datos: por ejemplo, si se cae durante un préstamo, al reiniciar el registro debe estar completo o no realizado, pero no a medio camino.
- Seguridad: Solo personal autorizado puede registrar o modificar préstamos y catálogo. Los usuarios socios podrían tener acceso limitado (por ejemplo, solo a consultar su propia cuenta o buscar libros, pero no a editar nada). Los datos personales de socios (dirección, teléfono, etc.) deben mantenerse privados según normativas de privacidad.
- Escalabilidad: Si la biblioteca se expande o pasa a formar parte de una red de bibliotecas, el sistema debería soportar múltiples sucursales o un mayor volumen de datos (ej., 50.000 libros). También podría contemplar extensiones futuras, como un módulo de reservas de libros (apartado) o gestión de multas, sin reestructurar completamente la arquitectura.

Detalles adicionales

- Tipo de usuario final: Bibliotecarios y personal administrativo (operan el sistema para préstamos, devoluciones, catálogo) y, potencialmente, usuarios de la biblioteca en modo consulta (para buscar libros o revisar el estado de sus préstamos, si se implementa un portal público).
- Entorno de uso: Computadores en el mostrador de préstamo y devolución, posiblemente equipados con lectores de código de barras para libros y carnés de socio (lo cual el sistema debe soportar como entrada de datos). Puede

haber un kiosko o PC de consulta para usuarios en sala. Conexión en red local en la biblioteca; opcionalmente acceso remoto si se habilita catálogo en línea.

- Tecnologías sugeridas: Aplicación cliente-servidor o web. Podría usarse un sistema web (Java, Python o PHP con base de datos MySQL) para facilitar varias estaciones conectadas. Integración con hardware: escáner de código de barras para ISBN y carnés (que actúa como entrada de teclado) y una impresora para recibos de préstamos si se requiere comprobante físico.
- Consideraciones: No se incluirá en esta versión la catalogación avanzada bajo estándares bibliotecarios (MARC, etc.) ni la generación de carnés físicos (se asume carnés existentes). El enfoque está en la operación diaria. La gestión de multas por retraso se manejará manualmente fuera del sistema (aunque este identifique préstamos atrasados).

PROYECTO 14
Sistema de gestión de voluntarios para una ONG

Descripción

La organización no gubernamental “Manos Amigas” coordina voluntarios para sus proyectos comunitarios (comedores sociales, campañas de recolección, etc.). Actualmente usan hojas de cálculo para listar voluntarios y asignarlos a actividades, comunicándose por correos masivos. Esto causa descoordinación: a veces hay más voluntarios de los necesarios o faltan en algunas actividades, y es difícil llevar registro de horas donadas o participación individual. Se propone un sistema de gestión de voluntarios que permita registrar a los voluntarios con sus preferencias, crear eventos o actividades, inscribir/asignar voluntarios a cada evento y registrar su asistencia y horas, mejorando la organización y reconocimiento del voluntariado.

Requerimientos funcionales

- RF1: Registrar voluntarios con sus datos de contacto, áreas de interés y disponibilidad. Criterio de aceptación: Al ingresar un nuevo voluntario (nombre, email/teléfono, habilidades o intereses, días disponibles), el sistema almacena su perfil sin duplicados (verifica email único). El voluntario queda disponible en la lista para ser asignado a actividades.
- RF2: Crear eventos o proyectos que requieran voluntarios, con sus detalles (descripción, fecha/hora, ubicación, número de voluntarios requeridos). Criterio de aceptación: Al registrar un nuevo evento con todos sus datos, este aparece en la lista de actividades abiertas, indicando la cantidad de voluntarios necesarios y cupos vacantes inicialmente iguales a ese número.
- RF3: Asignar voluntarios a eventos (o registrar su inscripción). Criterio de aceptación: El coordinador puede seleccionar un evento y ver voluntarios disponibles según sus preferencias/disponibilidad; al asignar un voluntario, el sistema lo vincula al evento y reduce el cupo disponible. Alternativamente, si los voluntarios pueden inscribirse por sí mismos (por invitación), el coordinador puede confirmar la inscripción en el sistema, logrando el mismo resultado. El sistema no debe permitir exceder el cupo definido.
- RF4: Registrar la asistencia y horas trabajadas de voluntarios en cada evento. Criterio de aceptación: Después de realizado el evento, el coordinador marca en el sistema qué voluntarios asistieron y cuántas horas contribuyeron cada uno. Esas horas se acumulan en el historial de cada voluntario. Por ejemplo, si un voluntario asignado faltó, se deja en cero sus horas para ese evento, mientras que quienes asistieron suman las horas correspondientes.
- RF5: Comunicar información y recordatorios a los voluntarios. Criterio de

aceptación: El sistema permite enviar un correo electrónico (o notificación) a todos los voluntarios asignados a un evento con detalles e instrucciones, y enviar recordatorios automáticos pocos días antes. Se verifica que, al activar un envío, todos los destinatarios reciban el mensaje en sus direcciones registradas.

Requerimientos no funcionales

- Usabilidad: Debe ser fácil de usar para coordinadores que pueden no ser expertos en tecnología. La navegación para registrar voluntarios o crear eventos debería ser sencilla, con formularios básicos. Si se habilita acceso a voluntarios (autogestión), la interfaz debe ser muy intuitiva y compatible con móviles, dado que los voluntarios podrían inscribirse desde sus teléfonos.
- Accesibilidad: Considerar diseño inclusivo, ya que voluntarios pueden ser de diversas edades y niveles técnicos. Uso de lenguaje claro y opción de ampliar texto o alto contraste ayudarían a usuarios con poca experiencia o dificultades visuales.
- Confiabilidad: La información sobre quién está asignado a cada evento no debe perderse ni duplicarse. Deben realizarse respaldos periódicos (por ejemplo, semanales) de la base de datos para proteger el historial de participación, útil para informes de impacto de la ONG (total de horas donadas, etc.).
- Seguridad: Los datos personales de voluntarios (contactos, posibles documentos) deben resguardarse; solo personal de la ONG con cuenta puede acceder al sistema. Al enviar correos masivos, debe evitarse exponer las direcciones de todos (usar copia oculta o envíos individuales automatizados). Además, cumplir con normativas de protección de datos (p. ej., GDPR) obteniendo consentimiento de voluntarios para almacenar sus datos y contactarles.
- Escalabilidad: Inicialmente se espera un número manejable de voluntarios (~100) y eventos esporádicos, pero el sistema debería soportar crecimiento (cientos de voluntarios, eventos concurrentes múltiples). También debería poder extenderse para nuevas funcionalidades, como un módulo de reportes detallados o integración con redes sociales para reclutar voluntarios, sin rehacer la base.

Detalles adicionales

- Tipo de usuario final: Coordinadores de voluntarios o personal de la ONG (administran el sistema) y eventualmente los propios voluntarios si se les da acceso para auto-inscripción (aunque inicialmente podría manejarse solo por

coordinadores).

- Entorno de uso: Computadoras en la oficina de la ONG para registro y planeación; acceso vía web desde cualquier lugar (posible trabajo remoto de coordinadores). Si voluntarios interactúan, lo harían desde sus dispositivos personales (PC o smartphones) a través de un portal web o aplicación dedicada.
- Tecnologías sugeridas: Aplicación web (intranet de la ONG) desarrollada con un framework común (Laravel, Django, etc.) y base de datos SQL. Integración con un servicio de correo (SMTP o API tipo SendGrid) para el envío de notificaciones. Un módulo web público sencillo (o enlaces por correo con token) podría implementarse si se desea que los voluntarios confirmen su asistencia en línea.
- Consideraciones: El sistema se enfoca en la gestión de personas y eventos de voluntariado; no abarca gestión de donaciones, financiamiento u otros aspectos operativos de la ONG. Se asume que la asignación de voluntarios es principalmente manual por parte de coordinadores al inicio, pero el diseño dejará abierta la posibilidad de un portal de autoservicio a futuro.

PROYECTO 15
Sistema de gestión de tickets de soporte técnico (Mesa de ayuda)

Descripción

La empresa de tecnología “TechSupport” ofrece servicio de soporte técnico a sus clientes y necesita organizar la atención de incidentes. Actualmente los clientes envían correos o llaman por teléfono, y los técnicos llevan un registro informal, lo que resulta en algunos casos olvidados o respuestas tardías. Se propone un sistema de mesa de ayuda que permita a los usuarios reportar incidencias mediante tickets, a los técnicos gestionar esos tickets asignándolos, actualizando su estado, y cerrándolos una vez resueltos, asegurando un seguimiento adecuado de cada solicitud.

Requerimientos funcionales

- RF1: Permitir a los usuarios reportar incidencias creando un ticket de soporte. Criterio de aceptación: Cuando un cliente/empleado llena un formulario de nuevo ticket (título, descripción del problema, categoría o prioridad), el sistema genera un identificador único, guarda la incidencia con estado inicial “Abierto”, y notifica al equipo de soporte sobre el nuevo ticket (p. ej., vía email).
- RF2: Asignar un ticket a un técnico de soporte. Criterio de aceptación: Un coordinador o el mismo técnico puede asignarse un ticket abierto; el sistema registra quién será responsable y cambia el estado a “En progreso”. El ticket muestra el nombre del técnico asignado y la fecha/hora de asignación.
- RF3: Actualizar el ticket con comentarios y cambiar su estado según el avance. Criterio de aceptación: El técnico puede agregar notas en el ticket (por ejemplo, “Se contactó al usuario para más detalles”) y modificar el estado según corresponda: “En espera” (si se espera información del cliente), “Resuelto” (si se dio una solución) etc. Cada actualización registra fecha y autor, y envía notificación al cliente si es relevante (p. ej., cuando se marca como Resuelto o se solicita información adicional).
- RF4: Notificar la resolución y cerrar el ticket. Criterio de aceptación: Al marcar un ticket como “Resuelto”, el sistema registra la fecha de resolución y envía un mensaje al cliente informando la solución o pasos realizados. Si el cliente confirma la solución o no responde en un tiempo definido, el soporte puede marcar el ticket como “Cerrado” definitivamente. El historial completo del ticket queda almacenado para consulta.
- RF5: Permitir la consulta y filtrado de tickets. Criterio de aceptación: Los

usuarios de soporte pueden listar tickets aplicando filtros por estado, por técnico asignado, por prioridad o por cliente. Por ejemplo, filtrar “Abiertos de alta prioridad” mostrará todos los tickets no resueltos marcados como alta prioridad. La lista resultante muestra campos clave (ID, título, cliente, estado, fecha de creación) ordenables para facilitar el seguimiento.

Requerimientos no funcionales

- Usabilidad: La plataforma debe ser intuitiva tanto para el cliente que reporta (formulario sencillo, campos claros) como para el técnico que gestiona (vista de tabla/resumen de tickets, con opciones para actualizar cada uno). Los clientes no expertos deben poder crear un ticket fácilmente, y los técnicos deben tener una vista eficiente de sus tareas.
- Notificaciones en tiempo real: Es importante que las partes involucradas reciban notificaciones oportunas. Cuando se crea o actualiza un ticket con información relevante para el cliente, este debe recibir un email inmediato. Los técnicos deberían ver nuevos tickets o cambios sin necesidad de refrescar (p. ej., mediante actualización automática o notificaciones push en la interfaz).
- Confiabilidad y disponibilidad: El sistema debe estar disponible siempre que se brinde soporte (idealmente 24/7 si hay soporte continuo). Debe tolerar picos de múltiples usuarios creando tickets a la vez sin fallar. Respaldos frecuentes de la base de datos garantizarán que no se pierda el historial de casos (lo cual es crítico para mantener la calidad y aprender de incidentes pasados).
- Seguridad: Los tickets pueden contener información sensible de la empresa o clientes; el acceso debe ser controlado. Un cliente solo puede ver sus propios tickets; los técnicos solo ven tickets de su ámbito o asignados (salvo quizá coordinadores que ven todo). Las credenciales de usuarios y comunicaciones deben estar cifradas (HTTPS) para proteger datos durante el uso remoto.
- Escalabilidad/Mantenibilidad: Si la base de clientes crece o se extiende el soporte a varios departamentos, el sistema debe soportar la división por colas o áreas sin perder desempeño. Además, el código debería ser modular para añadir funcionalidades futuras, como un módulo de base de conocimientos o métricas de desempeño (tiempos de resolución, satisfacción), sin requerir reescritura desde cero.

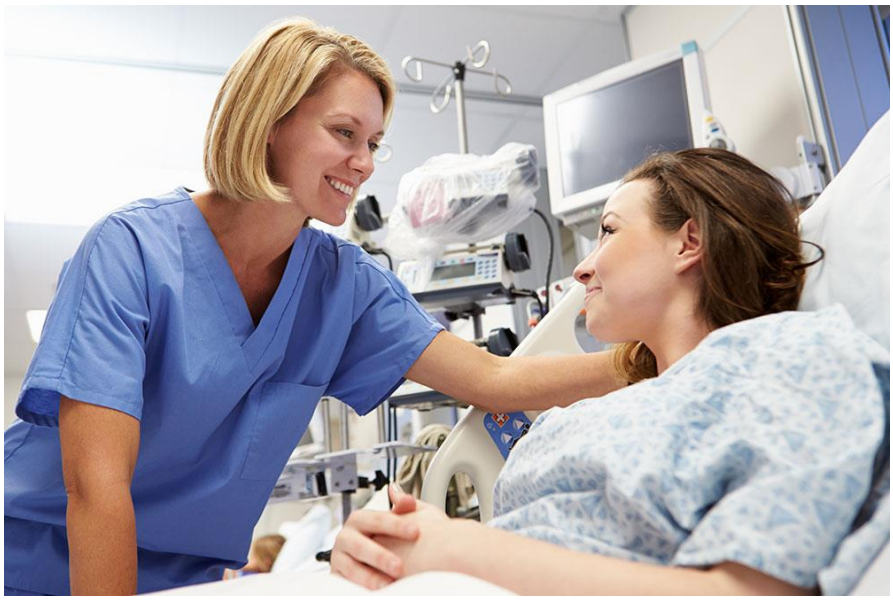
Detalles adicionales

- Tipo de usuario final: Usuarios finales que reportan problemas (clientes externos o empleados internos, dependiendo del contexto) y personal de

soporte técnico (agentes, coordinadores). Cada rol tiene permisos distintos en la aplicación.

- Entorno de uso: Aplicación accesible vía web: los clientes acceden desde un portal o formulario en internet/intranet para crear y ver sus tickets; los técnicos acceden desde sus PC de trabajo (o laptops) a la interfaz de gestión. Podría considerarse una aplicación móvil complementaria para técnicos on-call, pero la versión inicial se centrará en la versión web.
- Tecnologías sugeridas: Aplicación web con backend robusto (por ejemplo, Java Spring o Django) y base de datos relacional (PostgreSQL) para transacciones seguras. Interfaz web moderna (React, Angular, o plantilla Bootstrap) para proporcionar experiencia interactiva (actualizaciones dinámicas de la lista de tickets). Integración con servidor de correo para notificaciones. Posible uso de WebSockets para notificaciones en tiempo real en la interfaz de técnico.
- Consideraciones: No se implementará inicialmente una base de conocimientos ni encuestas de satisfacción post-cierre; el proyecto se enfoca en el flujo básico de ticketing. Se asume un volumen moderado (por ejemplo, una empresa con 200 empleados generando ~50 tickets al mes, atendidos por un equipo de 5 técnicos) como escenario de prueba, escalable posteriormente según necesidades reales.

PROYECTO 16
Sistema Integral de Internamiento y Altas de Pacientes de una Clínica Privada



Descripción del negocio

La clínica “San Gabriel Salud Integral” es una institución privada que ofrece servicios de hospitalización para pacientes con diversos diagnósticos médicos. Actualmente, el proceso de internamiento y alta se realiza en formularios manuales y planillas Excel, lo que genera errores administrativos, demoras en el acceso a camas disponibles, y falta de trazabilidad del historial de internamiento del paciente. Por ello, se propone el desarrollo de un Sistema Integral de Internamiento y Altas que permita gestionar el proceso clínico-administrativo de internación y egreso hospitalario, optimizando la asignación de camas, seguimiento del estado del paciente y generación de reportes para toma de decisiones.

Requerimientos funcionales del core business

- RF1: Registrar el ingreso de un paciente hospitalizado con sus datos clínicos, administrativos y cama asignada. Criterios de aceptación: Al ingresar los datos del paciente (nombre, edad, diagnóstico, médico tratante, fecha de ingreso, número de habitación y cama), el sistema crea un registro activo de internamiento y bloquea automáticamente la cama asignada, impidiendo que se duplique su uso.
- RF2: Visualizar en tiempo real el estado de ocupación de camas por piso o pabellón. Criterios de aceptación: Al consultar el módulo de disponibilidad, el usuario puede ver un mapa visual o listado de camas organizadas por

piso/pabellón, identificando cuáles están libres, ocupadas o en mantenimiento. La información debe estar actualizada al instante tras cada ingreso, traslado o alta.

- RF3: Registrar traslados de pacientes entre camas, habitaciones o servicios. Criterios de aceptación: Cuando el personal solicita un traslado, el sistema verifica disponibilidad en la nueva ubicación, actualiza la información del paciente y libera la cama anterior. Todo cambio queda registrado con fecha, hora y motivo del traslado.
- RF4: Registrar el alta médica de un paciente, cerrando su proceso de hospitalización. Criterios de aceptación: Al dar de alta a un paciente, el sistema solicita el motivo de egreso (alta médica, voluntaria, traslado, defunción), registra la fecha y hora de salida, libera la cama correspondiente y archiva el historial del internamiento para futuras consultas.
- RF5: Generar reportes de ocupación hospitalaria, duración de estancias y altas por período. Criterios de aceptación: Al seleccionar un rango de fechas, el sistema debe generar un informe con el número de ingresos, egresos, ocupación promedio diaria, y estancia promedio por paciente, exportable en PDF o Excel, y segmentado por servicio o especialidad.

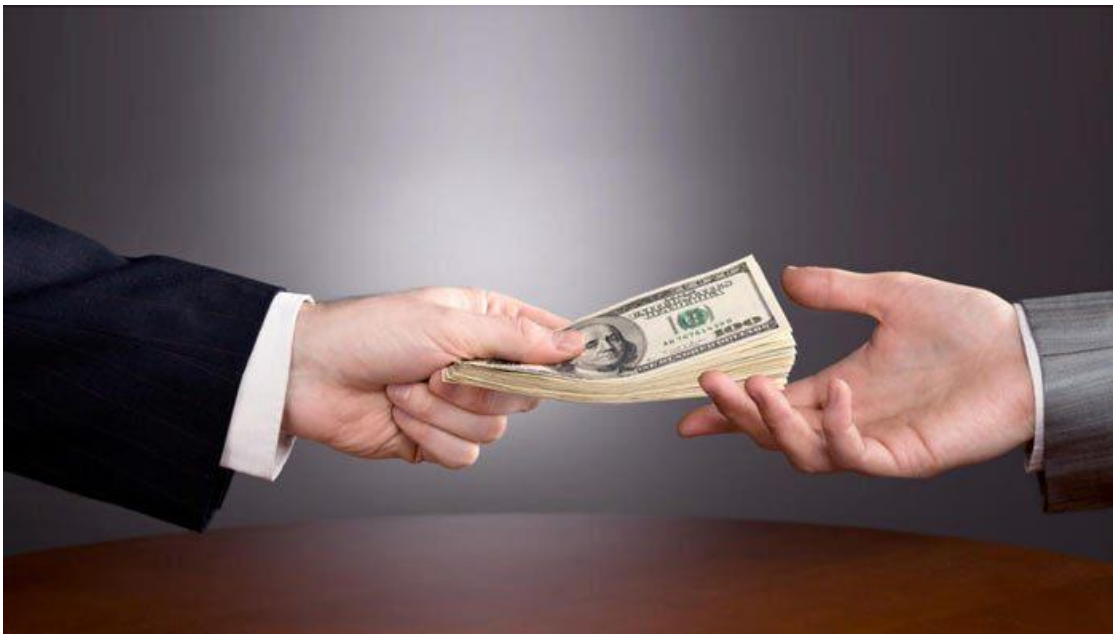
Requerimientos no funcionales

- Disponibilidad: El sistema debe estar disponible las 24 horas del día, los 7 días de la semana, considerando que los internamientos y altas pueden producirse en cualquier momento, incluyendo turnos nocturnos.
- Seguridad: El acceso debe estar protegido con autenticación y roles de usuario (por ejemplo, enfermería, médicos, personal administrativo). Los datos sensibles deben encriptarse y auditarse para cumplir normativas de confidencialidad (como la Ley de Protección de Datos Personales).
- Usabilidad: La interfaz debe ser intuitiva y accesible para el personal de salud con conocimientos básicos en informática. Debe incluir formularios simples, menús desplegables, autocompletado y colores diferenciados para los estados de camas (libre, ocupada, reservada, mantenimiento).
- Integridad de los datos: El sistema debe garantizar la no duplicidad de internamientos, validación de campos obligatorios y bloqueo de camas en uso, para asegurar que no se presenten errores críticos en la asignación de recursos.
- Escalabilidad: El sistema debe permitir la gestión de múltiples servicios o áreas clínicas (medicina, cirugía, pediatría, etc.) y poder crecer en volumen de pacientes y camas sin perder rendimiento. Su arquitectura debe permitir integrarse a futuro con sistemas complementarios como historias clínicas electrónicas o facturación.

Detalles adicionales para el equipo de desarrollo

- Usuarios previstos: Personal de admisión hospitalaria, enfermería, médicos tratantes, personal de estadísticas e informática.
- Tipo de acceso: Se espera acceso desde terminales en recepción y enfermería, además de una interfaz administrativa. En el futuro podría integrarse un acceso móvil restringido para médicos.
- Dispositivos utilizados: Computadoras de escritorio y laptops en red interna; se recomienda desarrollar una aplicación web para compatibilidad y acceso remoto controlado.
- Tecnologías sugeridas: Arquitectura cliente-servidor con backend en Java (Spring Boot) o Python (Django), base de datos relacional (PostgreSQL o MySQL), frontend responsivo en React o Angular, autenticación por roles, y generación de reportes en PDF o Excel.
- Restricciones de tiempo: Se espera que el proyecto pueda desarrollarse en un ciclo académico de 16 semanas, utilizando un enfoque incremental (p. ej., metodología AUP o SCRUM).
- Recomendaciones pedagógicas: Este proyecto permite a los estudiantes practicar la identificación de actores, modelado de procesos (casos de uso, diagramas de actividad), diseño de base de datos, interfaces y lógica de negocio con validaciones. También fomenta la reflexión ética sobre el manejo de información sensible en entornos de salud.

PROYECTO 17
Sistema de Gestión de Préstamos Personales para una Institución Financiera



Descripción del negocio

La institución financiera “Crédito Seguro” busca optimizar la gestión de sus préstamos personales mediante el desarrollo de una aplicación que automatice y controle eficientemente todo el ciclo de vida de los préstamos: desde la solicitud inicial, pasando por la evaluación crediticia y aprobación, hasta el seguimiento de pagos y cierre del préstamo.

Actualmente, estos procesos se realizan parcialmente con herramientas ofimáticas, lo que genera inconsistencias, riesgo de errores humanos y poca visibilidad sobre la cartera de préstamos y la morosidad.

El nuevo sistema deberá cumplir estrictamente las políticas crediticias internas del banco y los requisitos normativos sobre confidencialidad de datos financieros.

Requerimientos funcionales del core business

- RF1: Registrar solicitudes de préstamo incluyendo datos personales, laborales, ingresos, historial crediticio y documentos adjuntos del cliente. Criterios de aceptación: El sistema almacena la solicitud con todos los campos requeridos completados y genera un código único de seguimiento. Los archivos adjuntos quedan asociados a la solicitud y disponibles para consulta por el analista.

- RF2: Evaluar la capacidad de pago del cliente y asignar una calificación crediticia automática. Criterios de aceptación: El sistema aplica una fórmula predefinida sobre los ingresos, deudas y parámetros del banco. Muestra la capacidad de pago mensual estimada y una calificación (por ejemplo, A, B, C) basada en reglas internas. Solo se permite avanzar si la solicitud cumple con los criterios mínimos.
- RF3: Aprobar, rechazar o enviar a revisión las solicitudes según el análisis crediticio. Criterios de aceptación: El analista puede cambiar el estado de la solicitud a “Aprobado”, “Rechazado” o “En revisión”, registrando un comentario justificativo. Solo solicitudes con capacidad de pago suficiente pueden ser aprobadas.
- RF4: Generar el cronograma de pagos del préstamo una vez aprobado, incluyendo cálculo de intereses, cuotas y fechas de vencimiento. Criterios de aceptación: Dado el monto, tasa y plazo del préstamo, el sistema produce una tabla de amortización con los pagos mensuales detallados (interés, capital, cuota total). Este cronograma queda vinculado al préstamo aprobado y visible para seguimiento.
- RF5: Registrar y monitorear los pagos mensuales realizados por el cliente, incluyendo alertas por retraso. Criterios de aceptación: Cada vez que se registre un pago, el sistema actualiza el saldo pendiente, marca la cuota correspondiente como “pagada” o “vencida” según la fecha, y genera alertas internas si hay atraso mayor a 5 días.

Requerimientos no funcionales

- Seguridad de la información: Toda la información financiera y personal debe estar encriptada tanto en tránsito como en almacenamiento. Se implementarán políticas de control de acceso basadas en roles (analistas, supervisores, administradores).
- Trazabilidad y auditoría: El sistema debe registrar todo cambio significativo (estado de solicitud, pagos, ediciones) en un log con sello de tiempo, usuario responsable y detalle del cambio, para efectos de control interno.
- Disponibilidad: La aplicación debe estar disponible al menos en horario de oficina (8:00 a 18:00), y garantizar un 99% de disponibilidad, permitiendo consultas y registros sin interrupciones críticas.
- Usabilidad: El sistema debe facilitar el trabajo de los analistas de crédito con formularios intuitivos, validaciones claras y navegación simple. Se recomienda el uso de asistentes o indicadores visuales en el cronograma de pagos (colores para cuotas vencidas, próximas, etc.).
- Escalabilidad: Debe poder administrar una cartera de al menos 5,000 préstamos sin degradación del rendimiento. Además, su diseño modular debe

permitir la futura integración con sistemas contables o plataformas web de autoservicio para los clientes.

Detalles adicionales para el equipo de desarrollo

- Usuarios del sistema: Analistas de crédito, supervisores financieros, personal de caja (registro de pagos) y administradores del sistema. En versiones futuras, se puede contemplar un módulo web para clientes.
- Dispositivos de acceso: Estaciones de trabajo dentro del banco (aplicación web interna), con acceso controlado mediante credenciales.
- Tecnologías sugeridas: Backend en Java con Spring Boot o Python con Django, frontend con React o Angular, base de datos relacional PostgreSQL, SQL Server o MySQL. Cálculo de cuotas mediante el método francés de amortización.
- Componentes clave del sistema: Módulo de solicitud de préstamo, Evaluador de crédito automatizado, Generador de cronograma de pagos, Seguimiento de pagos e historial, Módulo de alertas y notificaciones, Panel de reportes financieros e indicadores.
- Tiempo estimado de desarrollo: Proyecto diseñado para ser desarrollado en 16 semanas académicas, aplicando metodología AUP o SCRUM con entregas iterativas. Se recomienda construir prototipos funcionales progresivos: Registro de solicitud, Evaluación crediticia, Cronograma de pagos, Registro de pagos, Reportes y alertas.
- Aspectos éticos y regulatorios: Los estudiantes deben discutir temas como la protección de datos personales, ética en decisiones de aprobación/rechazo de créditos, y cumplimiento con regulaciones bancarias (ej., verificación de identidad, prevención de fraude).