**Guía Informe final de proyecto de Desarrollo de SIA**

# Introducción

El control de asistencia es fundamental para una gestión eficiente del personal en cualquier empresa. Este proyecto propone el desarrollo de un sistema automatizado de toma de asistencia que permita registrar entradas y salidas de los empleados de forma precisa, reduciendo errores manuales y mejorando la eficiencia administrativa.

# Etapa 1: Visión y Alcance del Proyecto

1.1 Visión.

*Un sistema moderno de control de asistencia que simplifica el registro con códigos QR y mejora la gestión del tiempo y la información. Asistencia rápida, precisa y sin contacto.*

1.2 Problema y contexto

*Queremos resolver la dificultad y el tiempo que implica registrar manualmente la asistencia de personas en una organización, evitando errores humanos y mejorando la eficiencia del proceso.*

*En evidencia documentada se reportan deficiencias técnicas y operativas con los métodos manuales para toma de asistencia, registros ilegibles, incorrectos e incompletos, algunos como falsificación de datos o falta de respaldo. También se han reportado ineficiencias operativas como dificultad para realizar reportes.*

1.3 Usuarios y stakeholders

*Empleados: necesitan registrar su asistencia de forma rápida y sin errores. Supervisores: requieren controlar puntualidad y detectar ausencias fácilmente.*

*Recursos Humanos: busca automatizar el registro y reducir errores en nómina.*

*Gerencia: necesita datos confiables para tomar decisiones y mejorar productividad. Auditores: requieren registros claros y trazables.*

*Proveedores de software: necesitan un sistema compatible y fácil de mantener.*

1.4 Propuesta de valor

*El software permitirá registrar asistencias de forma rápida, segura y automatizada usando códigos QR, reduciendo tiempos de registro, evitando suplantaciones y permitiendo generar reportes confiables en tiempo real. El proyecto será exitoso si: el sistema puede escanear los QR y registrar asistencia de forma inmediata, se pueden generar reportes de asistencia por fecha, persona y hora, los usuarios valoran el sistema como más eficiente que el registro manual.*

# Etapa 2: Organización y Planificación del Proyecto

En esta fase, se debe realizar toda la organización administrativa del proyecto. Esto incluye la planificación de tareas, asignación de roles, creación de una carta Gantt, y selección de herramientas de gestión de proyectos. Este paso es crucial para asegurar que el proyecto avance de manera ordenada y eficiente.

2.1 Definición de Roles y Responsabilidades.

|  |  |  |
| --- | --- | --- |
| ROL | RESPONSABLE | RESPONSABILIDAD |
| Líder del proyecto | Sebastian Mejias | Coordina el equipo, realiza seguimiento del proyecto, comunica con stakeholders. |
| Desarrollador Backend | Paula Henriquez | Implementa lógica de negocio, base de datos y seguridad del sistema. |
| Desarrollador Frontend | Paula Henriquez | Diseña e implementa la interfaz de usuario. |
| Analista de Requisitos | Alison Ramirez | Recolecta requerimientos funcionales y no funcionales. |
| Tester / QA | Alison Ramirez | Verifica la calidad del software y realiza pruebas funcionales y no funcionales. |
| Administrador DevOps | Sebastian Mejias | Gestiona el entorno de despliegue, integración continua, backups, y monitoreo. |

* 1. Planificación de Tareas (Carta Gantt/ Kanban.).

|  |  |  |  |
| --- | --- | --- | --- |
| Fase | Tareas clave | Duración estimada | Encargado |
| Análisis de requisitos | Levantamiento de requerimientos | 1 semana | Analista de Requisitos |
| Diseño del sistema | Arquitectura, diagramas, prototipos UI | 1 semana | Desarrollador Frontend / Administrador DevOps |
| Desarrollo Backend | API, base de datos, seguridad | 2 semanas | Desarrollador Backend / Administrador DevOps |
| Desarrollo Frontend | UI, integraciones con API | 2 semanas | Administrador DevOps / Desarrollador Backend |
| Pruebas y QA | Testing, corrección de errores | 1 semana | Tester |
| Despliegue y documentación | Deploy, manuales, capacitación | 1 semana | Todo el equipo |

2.3 Herramientas de Organización y Gestión del Proyecto.

Para asegurar el cumplimiento de los objetivos del proyecto y facilitar la colaboración entre los miembros del equipo, se utilizarán diversas herramientas de organización y gestión. Estas permitirán planificar, asignar tareas, monitorear avances y garantizar la comunicación efectiva.

### a) Herramientas de Gestión de Tareas y Proyectos

* **Trello / Gira**: Plataformas basadas en tableros Kanban que permiten organizar las tareas en columnas (*Pendiente, En progreso, Completado*), asignarlas a responsables, establecer prioridades y fechas de entrega.
* **Click Up / Asana**: Alternativas para gestión ágil de proyectos, con integración de cronogramas, dependencias y seguimiento del progreso.

### b) Herramientas de Comunicación y Colaboración

* **Slack / Microsoft Teams**: Canales de mensajería y videollamadas para mantener una comunicación fluida en tiempo real entre los miembros del equipo y los interesados.
* **Google Workspace (Docs, Sheets, Drive)**: Para la creación, almacenamiento y edición colaborativa de documentos y reportes del proyecto.

### c) Herramientas de Control de Versiones

* **GitHub / GitLab**: Repositorios de código que permiten el trabajo colaborativo, el control de versiones y la integración continua, garantizando la trazabilidad de los cambios en el software.

### d) Herramientas de Seguimiento del Tiempo y Productividad

* **Clockify / Toggl**: Aplicaciones para registrar el tiempo invertido en cada tarea, facilitando la medición de productividad y la estimación de esfuerzo.

### e) Herramientas de Documentación Técnica

* **Notion / Confluence**: Plataformas para centralizar la documentación técnica, manuales, diagramas y decisiones de diseño, asegurando la accesibilidad de la información.

2.4 Matriz de riesgos inicial (seguridad, datos, continuidad).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | ***Riesgo identificado*** | ***Categoría (Seguridad/Datos/Continuidad)*** | ***Probabilidad (Alta/Media/Baja)*** | ***Impacto (Alto/Medio/Bajo)*** | ***Mitigación inicial*** | ***Responsable*** | | --- | --- | --- | --- | --- | --- | | *Acceso no autorizado a la BD* | *Seguridad* | *Media* | *Alto* | *Autenticación fuerte, cifrado, roles RBAC* | *Dev Backend* | | *Pérdida de información crítica* | *Datos* | *Media* | *Alto* | *Backups automáticos, control de versiones* | *DevOps* | | *Caída del servidor durante clase/demo* | *Continuidad* | *Alta* | *Medio* | *Infraestructura escalable en la nube, monitoreo* | *DevOps* | | *Fuga de datos sensibles de clientes* | *Seguridad* | *Baja* | *Alto* | *Cifrado en tránsito y en reposo, NDAs, políticas de acceso* | *Líder del proyecto* | | *Error humano al borrar registros* | *Datos* | *Media* | *Medio* | *Confirmaciones dobles, control de acceso, backups diarios* | *QA / Dev Backend* | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  |  |  |  | |

2.5 Checklist de decisión rápida (producto vs SaaS, nube vs on-premise, etc.).

|  |
| --- |
| 1. Producto instalado vs SaaS (Software como servicio) Producto instalado (On-Premise)  SaaS (Servicio en la nube)  Justificación: Evita costos iniciales de infraestructura, fácil mantenimiento, escalabilidad, y actualizaciones automáticas. 2. Infraestructura: Nube vs On-Premise On-Premise**.**  Nube (IaaS/PaaS/DBaaS)Justificación:Mayor flexibilidad, escalabilidad, reducción de costos operativos y alta disponibilidad. 3. Arquitectura: Monolito vs Microservicios/Servicios Monolito modular  Microservicios / SOAJustificación:Menor complejidad para el tamaño del proyecto, permite desarrollo ágil y simplifica mantenimiento inicial. 4. Proceso de desarrollo: Plan-Dirigido vs Ágil/DevOps Plan-dirigido (Cascada / V-Model)  Ágil / DevOps (Scrum, Kanban, XP)Justificación:Permite iteraciones rápidas, adaptación al cambio, integración continua y mejora incremental. |

# Etapa 3: Selección del Modelo de Desarrollo

En esta fase, deberás seleccionar el modelo de desarrollo de software que utilizarás para guiar el proyecto. Justifica tu elección basándote en las características del proyecto y las características de cada modelo.

3.1 Descripción del Proyecto

El proyecto consiste en el desarrollo de un **sistema de control de asistencia para empresas mediante QR**, orientado a gestionar de manera eficiente la presencia y puntualidad de sus empleados. El sistema busca reemplazar métodos manuales o sistemas obsoletos, ofreciendo una solución moderna, segura y accesible desde distintos dispositivos.

Entre sus principales funciones se consideran:

* Registro de asistencia y salida mediante escaneo de QR.
* Validación en tiempo real de la identidad del empleado.
* Reportes y estadísticas de asistencia y puntualidad.
* Accesibilidad multiplataforma (web y dispositivos móviles).
* Seguridad en el manejo de datos.

3.2 Modelos de Desarrollo Considerados (cascada, incremental, iterativo, ágil)

1. **Modelo en Cascada  
   -** Estructurado, con fases secuenciales (análisis, diseño, desarrollo, pruebas, implementación).  
   - Adecuado para proyectos con requerimientos muy claros y poco cambiantes.  
   - Desventaja: poca flexibilidad ante cambios.
2. **Modelo Incremental  
   -** Desarrollo en entregas parciales (módulos funcionales que se van sumando).  
   - Permite mostrar avances al cliente y obtener retroalimentación temprana.  
   - Requiere una planificación cuidadosa para integrar los incrementos.
3. **Modelo Iterativo  
   -** El producto se construye y mejora en versiones sucesivas, refinando requisitos y funcionalidades.  
   - Permite aprendizaje y adaptación conforme avanza el proyecto.  
   - Riesgo: requiere coordinación constante y revisiones frecuentes.
4. **Modelo Ágil (Scrum, Kanban, etc.)  
   -** Orientado a la colaboración, flexibilidad y entregas rápidas.  
   -Favorece la adaptación a cambios en los requerimientos.  
   - Requiere un equipo comprometido, comunicación continua y gestión dinámica.

3.3 Justificación del Modelo Seleccionado (Cómo este modelo se adapta al ciclo de vida del proyecto elegido)

Para este proyecto, el **modelo de desarrollo Ágil** resulta el más adecuado, principalmente bajo el marco de **Scrum**, debido a las siguientes razones:

* **Requerimientos cambiantes:** Al tratarse de un sistema nuevo, es probable que las empresas necesiten ajustes durante el desarrollo (nuevos reportes, mejoras de seguridad, funciones adicionales). El modelo ágil facilita estas adaptaciones.
* **Entrega temprana de valor:** Con sprints cortos, se pueden liberar versiones parciales (ejemplo: primero el módulo de registro QR, luego reportes, luego estadísticas), permitiendo a los usuarios empezar a utilizar el sistema sin esperar la versión final.
* **Retroalimentación constante:** Los clientes pueden probar cada incremento y dar sugerencias, lo que garantiza que el producto final cumpla con las expectativas.
* **Reducción de riesgos:** Al trabajar en iteraciones, los problemas se detectan y corrigen de manera temprana, evitando fallas críticas en etapas avanzadas.
* **Escalabilidad:** Si el sistema se expande (integración con nóminas, geolocalización, control biométrico), el modelo ágil permite añadir funcionalidades sin rehacer el proyecto completo.

En conclusión, el **modelo ágil** se adapta mejor al ciclo de vida de este proyecto porque combina **flexibilidad, entregas rápidas, colaboración activa y mejora continua**, asegurando que el sistema de control de asistencia mediante QR evolucione según las necesidades de las empresas.

# Etapa 4: Recolección y Análisis de Requerimientos

En esta fase, deberás identificar y documentar los requisitos del sistema. Estos pueden dividirse en requisitos funcionales (qué debe hacer el sistema) y requisitos no funcionales (cómo debe comportarse el sistema). Identificar y justificar la técnica que ocuparon para identificar los requisitos (Como documento Anexo deben entregar el [DER](about:blank))

4.1 Identificación de la o las técnicas para la recolección de los requerimientos.

4.2 Requisitos Funcionales y No funcionales.

4.3 Requisitos de seguridad y privacidad (ejemplo: cifrado, roles, logs).

4.4 Otros Requisitos (de actores, funcionales, interacción).

4.5 Priorización de Requerimientos.

# Etapa 5: Diseño del Sistema

En esta fase, debes diseñar la arquitectura general del sistema. Define los componentes principales y sus interacciones, así como la base de datos que soportará al sistema.

3.1 Diagrama de Flujo de Datos.

3.2 Modelo de Datos Semánticos.

3.3 Modelo de Dominio.

3.3.1 Diagrama de Clases.

3.4 Modelo de Objetos.

3.4.1 Diagrama de Objetos.

3.4.2 Diagrama de Secuencia.

3.4 Diagrama de Arquitectura del Sistema (monolítico modular / servicio).

3.5 Diseño de la Base de Datos (Modelo Relacional).

3.6 Diseño de la Interfaz de Usuario.

3.7 Esquema de seguridad: autenticación/autorización, cifrado, segregación de datos.

3.8 Política de acceso y roles.

# Etapa 4: Desarrollo e Implementación

En esta fase, el diseño del sistema se convierte en código ejecutable. Documenta las decisiones importantes que tomaste durante la implementación y describe cómo implementaste los principales módulos del sistema.

4.1 Lenguajes y Herramientas Utilizados.

4.2 Implementation de Módulos Principales.

* 1. Problemas y Soluciones Durante la Implementación.

4.4 Gestión de versiones (Git/CI/CD).

4.5 Políticas de backup en desarrollo (ej.: dumps automáticos de BD).

# Etapa 5: Pruebas y corrección de errores

En esta fase, debes probar el sistema para asegurarte de que cumple con los requisitos especificados. Esto incluye pruebas unitarias, pruebas de integración, pruebas del sistema completo y pruebas de aceptación.

5.1 Plan de Pruebas (unitarias, integración, siatema, aceptación)

5.2 Casos de prueba documentados.

5.3 Pruebas de seguridad: inyección SQL, manejo de errores, validación de datos.

5.4 Pruebas de recuperación: restauración de BD desde backup, medición de RTO/RPO.

5.4 Reporte y corrección de errores (Formato para registrar fallos y tiempo de reparación).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Información General del Proyecto   * Nombre del Proyecto: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ * Nombre del Software: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ * Fecha de Inicio del Registro: //\_\_\_\_ * Fecha de Fin del Registro: //\_\_\_\_ * Nombre del Estudiante/Grupo: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ * Nombre del Responsable: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **ID de Fallo** | **Fecha y Hora del Fallo** | **Descripción del Fallo** | **Condiciones Previas** | **Impacto del Fallo** | **Tiempo de Detección (minutos)** | **Tiempo de Reparación (minutos)** | **Estado de la Reparación** | **Comentarios Adicionales** | | 1 | **/**/\_\_\_\_ hh | Breve descripción del fallo | Ejemplo: Al intentar dividir por cero, el software se cierra inesperadamente | Alto | 5 | 15 | Reparado / Pendiente | Detalles adicionales si es necesario | | 2 | **/**/\_\_\_\_ hh | Breve descripción del fallo | Ejemplo: Error al marcar tarea como completada | Medio | 3 | 10 | Reparado / Pendiente | Detalles adicionales si es necesario | | 3 | **/**/\_\_\_\_ hh | Breve descripción del fallo | Ejemplo: Resultados incorrectos en la división de números | Alto | 2 | 5 | Reparado / Pendiente | Detalles adicionales si es necesario | |

# Etapa 6: Seguridad, Respaldo y Recuperación de Datos

En esta etapa se definen, implementan y documentan las medidas que garanticen la integridad, disponibilidad y confidencialidad de la información gestionada por el sistema. Se deben aplicar controles de seguridad como políticas de acceso, gestión de usuarios y cifrado de datos. Asimismo, se establecen y prueban las políticas de respaldo periódico de la base de datos y de los archivos críticos del sistema, definiendo su frecuencia, responsables y herramientas utilizadas. Finalmente, se deben diseñar y ejecutar mecanismos de recuperación que permitan restaurar la información en caso de fallos, incidentes de seguridad o pérdida de datos, midiendo los tiempos de recuperación (RTO) y los puntos de restauración alcanzados (RPO).

6.2 Plan de respaldo: frecuencia, herramientas utilizadas, validación de integridad.

6.3 Plan de recuperación: pasos para restaurar, responsables, tiempos medidos.

6.4 Evidencia de simulacros de restore.

### Conclusiones y Lecciones Aprendidas

* Cómo se cumplieron los RA del curso.
* Valor del sistema desarrollado para la organización.
* Próximos pasos de mejora.