# Cuadro resumen de la metodología RUP

|  |  |
| --- | --- |
| **Aspecto** | **Contenido** |
| **Definición** | RUP aplicado a **Desparches**: Plataforma digital para conectar usuarios con eventos/promociones locales mediante geolocalización, recomendaciones personalizadas y gestión integral para organizadores. Metodología iterativa e incremental para mitigar riesgos técnicos y comerciales. |
| **Características** | - **Basado en casos de uso**: 17 casos identificados (ej: CU-002 "Búsqueda geolocalizada"). - **Arquitectura centrada**: Microservicios (Node.js/Python), PostgreSQL geoespacial, AWS. - **Iterativo**: 8 iteraciones de 3 semanas priorizando funcionalidades core (ej: autenticación, pagos). - **Enfoque en calidad**: Métricas de usabilidad (tiempo de aprendizaje <5 min) y rendimiento (<2 seg en búsquedas). |
| **Etapas** | 1. **Inicio**: Alcance definido (eventos urbanos excluyendo rurales), Documento de Visión, riesgos identificados (ej: competencia con Eventbrite). 2. **Elaboración**: Prototipo arquitectónico validando geolocalización en entornos urbanos densos. 3. **Construcción**: Desarrollo incremental (React Native + microservicios), versión beta con pruebas de usuario. 4. **Transición**: Despliegue en AWS, manuales de usuario y técnico. |
| **Esquema gráfico** | Metodologia RUP | Ingenieria De Software ITSON |
| **Artefactos o entregables** | **1. Documento de Visión**  Define el propósito de Desparches como plataforma líder en eventos locales. Incluye alcance (áreas urbanas), stakeholders clave (usuarios, comercios) y objetivos estratégicos.  **2. Especificación de Requisitos (SRS)**  Detalla 17 requisitos funcionales (ej: búsqueda geolocalizada) y no funcionales (ej: <2 seg en respuestas). Base contractual para el desarrollo.  **3. Casos de Uso**  Describe las 17 interacciones clave del sistema (ej: CU-004 visualización de eventos, CU-013 recomendaciones). Incluye flujos principals  **4. Modelo de Análisis y Diseño**  Contiene diagramas UML (clases, secuencia) y arquitectura de microservicios. Define estructura modular y patrones de diseño.  **5. Prototipos**  Versiones en Figma (web/móvil) para validar UI/UX. Incluye mapa interactivo y flujo de reservas con usuarios reales.  **6. Plan de Desarrollo**  Cronograma de 8 iteraciones (3 semanas c/u), asignación de recursos (equipo, presupuesto) y gestión de riesgos.  **7. Documento de Arquitectura**  Especifica tecnologías (React Native, Node.js, AWS), vistas lógicas y de despliegue. Justifica decisiones técnicas clave.  **8. Modelo de Datos**  Describe entidades (Evento, Usuario), relaciones y esquema físico en PostgreSQL. Incluye normalización y claves.  **9. Casos de Prueba**  Pruebas derivadas de casos de uso (ej: CU-005 con tarjetas de prueba). Valida funcionalidad, rendimiento y seguridad.  **10. Bitácora de Cambios**  Control de versione, trazabilidad de modificaciones y artefactos.  **11. Manual de Usuario**  Guía paso a paso para reservar eventos y gestionar perfil. Incluye solución a errores comunes (GPS, pagos).  **12. Manual Técnico**  Instrucciones para despliegue en AWS, configuración de entornos y mantenimiento (backups, monitoreo).  **13. Informe de Evaluación**  Compara resultados con objetivos: 95% requisitos cumplidos, 4.5/5 en usabilidad y lecciones aprendidas**.** |
| **Ventajas** | - **Gestión de riesgos**: Identificación temprana de dependencia de APIs externas (Google Maps, Stripe). - **Trazabilidad**: CU-013 "Recomendaciones personalizadas" vinculado a requisitos y pruebas. - **Flexibilidad**: Adaptación para incluir gamificación en iteración 7. |
| **Desventajas** | - **Documentación extensa**: 15+ artefactos generados (ej: Bitácora de Cambios para 83 actualizaciones). - **Curva de aprendizaje**: Equipo requirió capacitación en React Native y microservicios. |