

**1. Modelo de Desarrollo Adoptado: SCRUM**

* **Iteraciones**: Sprints de 2 semanas.
* **Entregables**: Incrementos funcionales y testeados del sistema.
* **Eventos**: Planificación de sprint, daily meetings, revisiones, retrospectivas.

**2. Fases del Proyecto**

**Fase 1: Análisis y Diseño**

* **Objetivo**: Aterrizar los requerimientos y definir la arquitectura.
* **Actividades**:
  + Revisión y validación de RF y RNF con stakeholders.
  + Diagramación de flujos con Figma, Lucidchart, Draw.io.
  + Diseño de UI/UX con Figma, Adobe XD, Whimsical.
  + Diseño de arquitectura técnica (microservicios, seguridad, disponibilidad).
* **Entregables**:
  + Mockups de pantallas.
  + Diagramas de flujo de navegación y arquitectura.
  + Documentación de requisitos en Notion/Markdown.

**Fase 2: Desarrollo**

* **Objetivo**: Construir los módulos funcionales siguiendo las prioridades de impacto y complejidad.
* **Actividades**:
  + Configuración de ambientes de desarrollo y CI/CD.
  + Implementación por módulos (seguridad, cursos, usuarios, evaluaciones…).
  + Integración de servicios (Firebase, Moodle, Auth0, Keycloak, etc.).
* **Priorización**:
  + Alta prioridad: RF1, RF2, RF3, RF4, RF14, RF15, RNF1, RNF2.
  + Baja prioridad: RF16–RF22 (post-MVP).
* **Herramientas**: GitHub Projects, Trello (seguimiento), Jira.

**Fase 3: Pruebas y Validación**

* **Objetivo**: Asegurar la calidad funcional, rendimiento, seguridad y usabilidad.
* **Actividades**:
  + Pruebas unitarias y de integración (Jest, PyTest, PHPUnit, según stack).
  + Pruebas de rendimiento (JMeter, LoadNinja).
  + Pruebas de seguridad (OWASP ZAP, Burp Suite).
  + Usabilidad (Maze, UserTesting, Hotjar).
* **Entregables**:
  + Informes de pruebas automatizadas.
  + Reporte de métricas de carga y seguridad.

**Fase 4: Implementación y Despliegue**

* **Objetivo**: Publicar la versión estable del sistema.
* **Actividades**:
  + Configuración de infraestructura (AWS, GCP, Kubernetes).
  + Configuración de redundancia y escalabilidad.
  + Automatización de backups y monitoreo.
  + Publicación en producción.
* **Entregables**:
  + Sistema disponible 24/7.
  + Documentación técnica (Swagger, Markdown).

**Fase 5: Mantenimiento y Mejora Continua**

* **Objetivo**: Dar soporte post-lanzamiento y evolucionar el producto.
* **Actividades**:
  + Soporte técnico (Zendesk, Intercom).
  + Gestión de incidencias (Jira, GitHub Issues).
  + Incorporación de feedback de usuarios.
  + Revisión periódica de rendimiento y seguridad.

| Área | Herramienta / Tecnología |
| --- | --- |
| Lenguaje Frontend | JavaScript (ES6+) |
| Framework Principal | Spring Boot |
| Estilos | Tailwind CSS / Styled Components |
| Manejo de Estados | Redux Toolkit / React Context |
| Backend | Java 21 + Springboot |
| Base de Datos | PostgreSQL (SQL) |
| Control de Versiones | Git + GitHub |
| Gestión de Proyecto | Trello / Jira con tableros Kanban |
| Entorno de Desarrollo | Visual Studio Code |
| Testing | Postman |
| Integración Continua | GitHub |
| Despliegue | Vercel / Netlify para frontend |
| Modelado UML | Draw IO |

Para aplicar el modelo se diseñará:

| **Vista del modelo 4+1** | **Diagramas UML recomendados** |
| --- | --- |
| **Vista Lógica** | Diagrama de Clases |
| **Vista de Implementación** | Diagrama de Componentes |
| **Vista de Procesos** | Diagrama de Actividades |
| **Vista de Despliegue** | Diagrama de Despliegue |
| **Vista de Escenarios (Casos de Uso)** | Diagrama de Casos de Uso |

**PATRÓN DE DISEÑO DE SOFTWARE / ARQUITECTURA DE DISEÑO**

Definir entre:

Patrón por capas.

MVC

Microservicios