# Portada

**Proyecto**: Maqueta interactiva para plataforma de gestión de cursos  
**Versión**: 1.0  
**Fecha**: 01‑09‑2025  
**Autoría**: Jorge Salgado

# Resumen Ejecutivo

* La propuesta describe una maqueta interactiva de una plataforma de gestión de cursos diseñada para administradores, contratistas y usuarios finales.
* Se basa en tecnologías web modernas (React/Next.js, TypeScript, Tailwind CSS) y almacena datos de forma local para simular un backend.
* El objetivo es presentar un flujo completo: creación de cursos, inscripción masiva o manual, validaciones de capacidad y RUT, registro de asistencia y notas, generación de certificados y reportes gráficos.
* Se define una arquitectura ligera con capas de UI, dominio y servicios simulados que separan responsabilidades y facilitan pruebas.
* El diseño UX/UI utiliza principios de accesibilidad (WCAG 2.2 AA), diseño responsivo y patrones de calendario y tablas accesibles.
* El plan de trabajo de dos semanas abarca análisis de requisitos, prototipado de pantallas clave, integración de librerías (FullCalendar, SheetJS, PDF‑Lib, Recharts) y validaciones.
* Los KPIs de éxito incluyen cobertura del 100 % de requisitos en la maqueta, cumplimiento de accesibilidad, tiempos de carga inicial <3 s y generación correcta de certificados y reportes.
* Se identifican riesgos (sobrecarga de validaciones, complejidad de la UI, rendimiento) y medidas de mitigación (prototipos iterativos, uso de hooks y librerías optimizadas, pruebas automatizadas).

# Contexto y Alcance

## Cliente y producto (maqueta)

La maqueta se destina a un organismo de capacitación que gestiona cursos presenciales y virtuales. Busca visualizar la futura plataforma de formación sin desarrollar un backend real.

### Perfiles de usuario

| Perfil | Descripción |
| --- | --- |
| Administrador | Crea y gestiona cursos, carga asistencia y calificaciones, genera certificados y visualiza reportes. |
| Contratista | Inscribe participantes en cursos mediante calendario y carga masiva o manual; valida capacidad y RUT. |
| Usuario individual | Accede a sus datos, visualiza cursos inscritos y descarga certificados propios. |

### Módulos principales

| Módulo | Descripción resumida | Perfil(es) |
| --- | --- | --- |
| **Acceso** | Pantalla inicial con ingreso de usuario “robot” y clave que determina el perfil. | Todos |
| **Administración de cursos** | Creación y edición de cursos (nombre, código, duración, fechas, modalidad presencial/Teams, objetivos). | Administrador |
| **Carga de nómina** | Calendario/grilla de cursos; selección de día y carga de participantes con butacas (Teams=200, presencial=30); carga manual y por Excel con validaciones. | Contratista |
| **Asistencia y calificaciones** | Registro de asistencia y notas mediante carga de planillas; cálculo de porcentajes y estado aprobado/reprobado. | Administrador |
| **Certificados** | Generación masiva de certificados PDF para un curso y descarga individual para cada usuario. | Administrador, Usuario |
| **Reportes** | Grilla PDF y dashboard de gráficos con tendencias de asistencia y aprobación a lo largo del tiempo. | Administrador |

### Supuestos y exclusiones

* La maqueta no persiste datos en servidores; se usa almacenamiento local (IndexedDB/localStorage) sólo para la demo.
* No se realiza autenticación real ni se conecta a sistemas externos; el usuario “robot” es una convención.
* Las validaciones de RUT, capacidad y participación son simuladas; se incluyen algoritmos y mensajes de error, pero no consultas a servicios gubernamentales.
* No se aborda la emisión real de certificados con firmas digitales ni el envío de correos; se simula la generación de PDF para descarga.
* Las fechas se expresan en formato DD‑MM‑YYYY y se asume zona horaria America/Santiago.
* La maqueta se orienta a demostración y pruebas de UX; la lógica de negocio se estructura para ser migrada a un backend real en versiones futuras.

# Objetivos y KPIs

| Objetivo | Métrica | Meta a 2 semanas | Método de medición |
| --- | --- | --- | --- |
| Validar el modelo de negocio y los flujos de capacitación | Cobertura de requisitos funcionales | ≥ 100 % de requisitos trazados | Revisión de matriz de trazabilidad y walkthrough por perfiles. |
| Garantizar una experiencia accesible y responsiva | Cumplimiento de WCAG 2.2 nivel AA | 100 % de criterios clave | Auditoría con checklist y pruebas de navegación con teclado y lectores. |
| Simular procesos críticos (inscripción, asistencia, certificados) | Flujos navegables sin errores | 100 % de flujos habilitados | Pruebas E2E de inscripción masiva, generación de certificados y reportes. |
| Demostrar rendimiento aceptable en la maqueta | Tiempo de carga inicial en dispositivo estándar | < 3 s | Medición con herramientas de desarrollo y repetición en distintos navegadores. |
| Proporcionar documentación técnica verificable | Documento técnico‑funcional entregado | 1 documento consolidado | Evaluación por stakeholders de completitud y claridad. |

# Requisitos Funcionales

Los requisitos se descomponen en items atómicos (REQ‑X) agrupados por perfil y módulo. Cada historia sigue el patrón INVEST y cada criterio de aceptación se describe en formato Gherkin.

## Administrador

### Módulo de acceso

| ID | Historia de usuario (INVEST) | Criterios de aceptación (Gherkin) |
| --- | --- | --- |
| REQ‑A1 | **Como Administrador** quiero ingresar con usuario “robot” y una clave para acceder a mi panel de gestión de cursos, de modo que sólo quienes poseen la clave adecuada puedan gestionar los cursos. | **Dado** que estoy en la página de acceso **y** soy administrador **Cuando** ingreso la clave correspondiente **Entonces** la maqueta me redirige al panel de administración **Y** se visualizan las opciones de gestión de cursos, asistencia, certificados y reportes. |

### Administración de cursos

| ID | Historia de usuario (INVEST) | Criterios de aceptación (Gherkin) |
| --- | --- | --- |
| REQ‑A2 | **Como Administrador** quiero crear un curso indicando nombre, código, duración, fechas de impartición, modalidad (presencial o Teams) y objetivos, para que la oferta esté actualizada. | **Dado** que accedo al módulo de administración **Cuando** presiono “Nuevo curso” **Entonces** se muestra un formulario vacío **Y** se valida que los campos obligatorios estén completos **Y** el código sea único. **Cuando** guardo un curso válido **Entonces** la grilla de cursos se actualiza con el nuevo registro. |
| REQ‑A3 | **Como Administrador** quiero editar o eliminar cursos existentes para corregir información o desactivar cursos obsoletos. | **Dado** un curso ya creado **Cuando** selecciono “Editar” **Entonces** se cargan sus datos en el formulario **Y** puedo modificar cualquier campo salvo el código. **Cuando** confirmo cambios **Entonces** la grilla se actualiza. **Cuando** presiono “Eliminar” **Entonces** se pide confirmación **Y** al aceptar se elimina el curso de la lista. |

### Asistencia y calificaciones

| ID | Historia de usuario (INVEST) | Criterios de aceptación (Gherkin) |
| --- | --- | --- |
| REQ‑A4 | **Como Administrador** quiero cargar una planilla de asistencia y calificaciones para un curso seleccionado, de modo que el sistema calcule porcentajes y determine si cada participante aprueba o reprueba. | **Dado** que selecciono un curso en el módulo de asistencia **Cuando** importo un archivo Excel válido con RUT y notas **Entonces** se calcula para cada alumno el porcentaje de asistencia (según nº de sesiones) **Y** se calcula el estado final (aprobado/reprobado) según reglas definidas (ej. ≥ 50 % asistencia y nota ≥ 4,0). **Si** el Excel contiene RUT inválidos **Entonces** se muestran mensajes de error específicos antes de guardar. |
| REQ‑A5 | **Como Administrador** quiero visualizar la lista de participantes con sus porcentajes y notas para auditar las asistencias y realizar correcciones antes de cerrar el curso. | **Dado** un curso con planilla cargada **Cuando** visualizo la grilla de asistencia **Entonces** cada fila muestra nombre, RUT, porcentaje de asistencia, nota y estado. **Cuando** edito manualmente un valor **Entonces** la maqueta recalcula el estado automáticamente **Y** advierte si se incumple algún requisito mínimo. |

### Certificados y reportes

| ID | Historia de usuario (INVEST) | Criterios de aceptación (Gherkin) |
| --- | --- | --- |
| REQ‑A6 | **Como Administrador** quiero generar y descargar certificados en lote para un curso, de forma que pueda entregar comprobantes a todos los participantes aprobados. | **Dado** un curso con participantes aprobados **Cuando** selecciono la opción “Generar certificados” **Entonces** la maqueta produce un PDF por participante con datos como nombre, RUT, nombre del curso, horas y estado **Y** ofrece un archivo ZIP descargable con todos los certificados. **Si** no hay participantes aprobados **Entonces** se muestra un mensaje informando que no es posible generar certificados. |
| REQ‑A7 | **Como Administrador** quiero ver reportes de asistencia y aprobación mediante tablas y gráficos para analizar tendencias y tomar decisiones. | **Dado** que existen datos de múltiples cursos **Cuando** accedo a la sección de reportes **Entonces** se presenta un dashboard con indicadores clave: porcentaje de asistencia promedio, tasa de aprobación y gráfica de evolución por periodo. **Además** puedo filtrar por curso, contratista o rango de fechas. **Los reportes** permiten exportar a PDF o imagen para compartir. |

## Contratista

### Módulo de acceso

| ID | Historia de usuario (INVEST) | Criterios de aceptación (Gherkin) |
| --- | --- | --- |
| REQ‑C1 | **Como Contratista** quiero iniciar sesión con mi clave para acceder al calendario de cursos y poder inscribir participantes. | **Dado** que estoy en la pantalla de acceso **Cuando** ingreso la clave de contratista **Entonces** la maqueta me redirige al calendario de cursos. |

### Calendario y carga de nómina

| ID | Historia de usuario (INVEST) | Criterios de aceptación (Gherkin) |
| --- | --- | --- |
| REQ‑C2 | **Como Contratista** quiero visualizar un calendario/grilla con todos los cursos disponibles diferenciando tipo (propios u otros) mediante códigos de color. | **Dado** que accedo al calendario **Entonces** cada curso se muestra con color según su tipo (propio u externo) **Y** cada tarjeta incluye nombre, código, horario (mañana/tarde) y modalidad. **Los cursos del día actual** destacan visualmente para facilitar la selección. |
| REQ‑C3 | **Como Contratista** quiero seleccionar un curso y un día específico para abrir la pantalla de carga de participantes con visualización de butacas disponibles. | **Dado** un calendario con cursos **Cuando** hago clic en un curso **Entonces** se despliega un calendario mensual resaltando los días de dictado. **Cuando** selecciono un día disponible **Entonces** la maqueta muestra una interfaz de butacas (capacidad: 200 en Teams y 30 presencial) con los lugares ocupados/libres. |
| REQ‑C4 | **Como Contratista** quiero inscribir participantes manualmente completando campos (nombre completo, RUT, contratista) para cada butaca. | **Dado** la pantalla de butacas **Cuando** selecciono una butaca libre **Entonces** se abre un formulario para ingresar nombre, RUT y contratista. **El sistema** valida el formato de RUT, verifica capacidad y detecta si el RUT ya está inscrito en otro curso o sesión. **Cuando** guardo el participante **Entonces** la butaca se marca como ocupada y la lista se actualiza. **Si** hay errores (RUT inválido, duplicado) **Entonces** se muestran mensajes bajo los campos correspondientes. |
| REQ‑C5 | **Como Contratista** quiero cargar una nómina masiva mediante un archivo Excel con RUTs y nombres para agilizar la inscripción de grupos completos. | **Dado** que estoy en la pantalla de carga masiva **Cuando** selecciono un archivo Excel con columnas predefinidas (Nombre, RUT, Contratista) **Entonces** la maqueta lee el archivo usando SheetJS, detecta filas con errores (RUT inválido, nombre no coincide) y muestra un resumen con número de inscripciones exitosas y fallidas. **Cuando** confirmo la carga **Entonces** las butacas se actualizan, sin superar la capacidad. |
| REQ‑C6 | **Como Contratista** quiero recibir alertas claras cuando excedo la capacidad, intento inscribir participantes repetidos o con RUT incorrecto. | **Dado** que registro participantes manual o masivamente **Cuando** el número de inscritos supera la capacidad **Entonces** se muestra un mensaje de error y se impide la inscripción. **Cuando** un RUT ya está inscrito en el curso o en otro curso **Entonces** se alerta y se solicita corrección. **Cuando** el dígito verificador del RUT no coincide **Entonces** se indica el error y se evita guardar el registro. |

## Usuario individual

| ID | Historia de usuario (INVEST) | Criterios de aceptación (Gherkin) |
| --- | --- | --- |
| REQ‑U1 | **Como Usuario** quiero ingresar a la plataforma con mi RUT y clave para ver mis cursos inscritos, mis calificaciones y descargar mis certificados aprobados. | **Dado** la pantalla de acceso **Cuando** ingreso mi clave de usuario **Entonces** accedo al panel personal donde se listan mis cursos aprobados y reprobados con fecha y modalidad. **Si** tengo cursos aprobados **Entonces** puedo descargar mi certificado individual en PDF. **Cuando** presiono “Descargar” **Entonces** se genera el PDF en mi navegador. |

# Arquitectura de la Solución (Maqueta)

## Diagrama textual de capas

┌─────────────────────────────────────────────┐  
│ Presentación (UI) │  
│ │  
│ • React/Next.js con TypeScript │  
│ • Componentes: Calendario, Grilla, Formulario│  
│ • Diseño responsive y accesible (WCAG 2.2) │  
└─────────────────────────────────────────────┘  
 ↓ eventos/props/estado  
┌─────────────────────────────────────────────┐  
│ Capa de Dominio (Servicios) │  
│ │  
│ • Servicios simulados (API fake) │  
│ – end points REST para cursos, sesiones, │  
│ inscripciones, asistencia, certificados │  
│ • Validaciones de negocio (capacidad, RUT) │  
│ • Manejo de Excel (SheetJS) y PDF (PDF‑Lib) │  
└─────────────────────────────────────────────┘  
 ↓ almacenamiento  
┌─────────────────────────────────────────────┐  
│ Persistencia Mock (Infra) │  
│ │  
│ • IndexedDB + Dexie para datos temporales │  
│ • localStorage para preferencias de usuario │  
│ • Generación de archivos (blob/URL) │  
└─────────────────────────────────────────────┘

graph LR  
UI[Presentación (UI)] --> Domain[Capa de Dominio (Servicios)]  
Domain --> Infra[Persistencia Mock (Infra)]

La arquitectura sigue principios de Clean Architecture: la UI se comunica con servicios de dominio a través de funciones asincrónicas, y estos a su vez gestionan los datos en IndexedDB. Las validaciones y lógica de negocio residen en la capa de dominio para facilitar su reutilización cuando se migre a un backend real.

## Stack recomendado

* **Framework**: Next.js 14 con App Router en modo SPA para simplificar despliegue; permite SSR si se requiere.
* **Lenguaje**: TypeScript para tipado fuerte y mejor DX.
* **UI**: React con Tailwind CSS y componente shadcn/ui para accesibilidad y personalización.
* **Estado**: Zustand para manejo de estado local; React Query si se simula interacción con API.
* **Calendario**: FullCalendar React para calendarios interactivos, con soporte de accesibilidad y eventos customizables.
* **Tablas**: TanStack Table para grillas reactivas y accesibles.
* **Formularios y validación**: React Hook Form + Zod para esquema de validación y mensajes.
* **Excel**: SheetJS (xlsx) para leer archivos Excel en cliente.
* **PDF**: PDF‑Lib o jsPDF para generación de certificados y reportes.
* **Gráficos**: Recharts, que soporta renderización accesible y se integra con React.
* **Persistencia mock**: IndexedDB con Dexie para almacenar cursos, sesiones, inscripciones y certificados de manera local y sin bloqueo.
* **Testing**: Playwright para pruebas E2E de los flujos críticos y Vitest para pruebas unitarias de validaciones.
* **Despliegue**: Vercel o entorno similar para mostrar la maqueta.

### Alternativas

| Categoría | Opción alternativa | Pros | Contras | Criterios de elección |
| --- | --- | --- | --- | --- |
| Framework UI | Vue 3 + Vite + Pinia | Curva de aprendizaje baja y sintaxis expresiva; buena DX; | Comunidad más pequeña que React; menos maduro para libs como FullCalendar. | Elegir si el equipo domina Vue o busca performance de Vite. |
|  | SvelteKit | Sencillez en la escritura y optimización automática; | Ecosistema más pequeño; integraciones a librerías específicas pueden requerir adaptadores. | Elegir si se prioriza performance y simplicidad en la maqueta. |
| Calendario | react-big-calendar | Sin dependencias de jQuery; integración simple con React; | Menos opciones de accesibilidad y personalización avanzada. | Elegir si se necesita un calendario básico y liviano. |
| PDF | jsPDF | Librería ligera y simple para documentos básicos; | Limitada para documentos complejos; menos soporte de imágenes vectoriales. | Elegir para certificados sencillos sin gráficos complejos. |
| Gráficos | Chart.js | Popular, fácil de usar, admite gráficos diversos; | API no nativa de React, se requiere wrapper; accesibilidad limitada por defecto. | Elegir si se necesitan gráficos sencillos y se prioriza soporte. |
| Persistencia | localStorage | Sencillo de usar, API sincrónica; ideal para pequeñas demos; | Capacidad limitada; no soporta estructuras complejas ni transacciones. | Elegir para pruebas rápidas o datos no estructurados. |

# Diseño UX/UI

## Arquitectura de información (IA)

La plataforma se estructura en cinco secciones principales accesibles según el perfil:

1. **Inicio de sesión**: pantalla común con campos de usuario y clave.
2. **Panel administrador**: enlaces a “Cursos”, “Asistencia y notas”, “Certificados” y “Reportes”.
3. **Panel contratista**: visualización de calendario y acceso a carga manual o masiva de nómina.
4. **Panel usuario**: listado de cursos inscritos con opción de descarga de certificados individuales.
5. **Soporte y ayuda**: sección persistente con accesos a documentación y contacto, disponible en todas las vistas conforme al criterio 3.2.6 de WCAG 2.2 sobre ayuda consistente【503959695386811†L246-L276】.

## Flujos de usuario (ASCII)

### Administrador – Gestión de cursos y asistencia

[Inicio de sesión]  
 ↓ (A1)  
[Dashboard administrador]  
 ↓  
┌───────────────┬─────────────────┬───────────────────┐  
│Cursos (A2-A3) │Asistencia (A4-A5)│Reportes (A7) │  
└───────────────┴─────────────────┴───────────────────┘  
 ↓  
 Cursos → Nueva/Editar curso → Guardar/Eliminar  
 Asistencia → Seleccionar curso → Cargar Excel → Recalcular estados  
 Reportes → Filtros → Ver tablas y gráficos  
 ↓  
 Certificados (A6) → Seleccionar curso → Generar PDF/ZIP

graph TD  
A[Inicio de sesión] --> B[Dashboard administrador]  
B --> C[Cursos]  
B --> D[Asistencia]  
B --> E[Reportes]  
C --> F[Nueva/Editar curso]  
D --> G[Seleccionar curso]  
G --> H[Cargar Excel]  
E --> I[Filtros]  
I --> J[Ver tablas y gráficos]  
B --> K[Certificados]  
K --> L[Seleccionar curso]  
L --> M[Generar PDF/ZIP]

### Contratista – Inscripción en calendario

[Inicio de sesión]  
 ↓ (C1)  
[Calendario de cursos]  
 ↓  
 Seleccionar curso (C2)  
 ↓  
 Ver días disponibles (C3)  
 ↓  
 Seleccionar día → Ver butacas  
 ↓  
 ┌─────────────┬─────────────────┐  
 │Carga manual │Carga masiva (C5)│  
 └─────────────┴─────────────────┘  
 ↓  
 Registrar participantes (C4/C5) con validaciones (C6)  
 ↓  
 Confirmar inscripción

graph TD  
A[Inicio de sesión] --> B[Calendario de cursos]  
B --> C[Seleccionar curso]  
C --> D[Ver días disponibles]  
D --> E[Seleccionar día]  
E --> F[Ver butacas]  
F --> G[Carga manual]  
F --> H[Carga masiva]  
G --> I[Registrar participantes]  
H --> I

### Usuario individual – Descarga de certificados

[Inicio de sesión]  
 ↓ (U1)  
[Panel personal]  
 ↓  
 Ver cursos aprobados/reprobados  
 ↓  
 Descargar certificado de cursos aprobados

graph TD  
A[Inicio de sesión] --> B[Panel personal]  
B --> C[Ver cursos]  
C --> D[Descargar certificado]

## Design system

### Tokens

* **Tipografía**: se emplean escalas modulares con fuentes sans‑serif legibles (por ejemplo, Inter o Roboto). Tamaños base 16 px, con jerarquía h1 2 rem, h2 1,5 rem, h3 1,25 rem y texto 1 rem.
* **Colores**: paleta con contraste elevado (ratio ≥ 4.5:1).
  + Primario: azul oscuro para barras de navegación y botones destacados (#0A3D62).
  + Secundario: verde (#4CAF50) para estados aprobados y confirmaciones.
  + Advertencia: naranja (#F57C00) para alertas de capacidad y validaciones.
  + Error: rojo (#D32F2F) para RUT inválido o duplicado.
  + Fondo claro: gris muy claro (#F9FAFB) que facilita el contraste.
* **Espaciado**: escala basada en 4 px (4, 8, 16, 24, 32) para márgenes y padding.
* **Radios**: bordes suaves de 4 px en tarjetas y formularios.
* **Sombras**: sombras sutiles para separar tarjetas del fondo, siguiendo un esquema de elevación de 1 a 3 niveles.
* **Iconografía**: se usan íconos de librerías accesibles (Heroicons) como complementos, no como elementos independientes; cada icono incluye atributo aria-label.

### Componentes base

1. **Tabla**: utiliza TanStack Table; incluye caption descriptiva, cabeceras marcadas con <th> y scope, filas alternadas (zebra) y resaltado de fila/columna al pasar el mouse o enfocar con teclado. Se sigue la guía WAI de mantener estructuras simples, alinear texto a la izquierda y datos numéricos a la derecha【484670439075094†L94-L146】.
2. **Calendario**: implementado con FullCalendar; permite vista mensual y agenda; soporta navegación con teclado (flechas para días, PageUp/PageDown para meses) y muestra formato de fecha. Permite entrada manual de fechas; se evita la auto‑sumisión para no interrumpir lectores de pantalla【349603758647189†L301-L334】.
3. **Formulario**: campos con label asociados, placeholders claros y mensajes de ayuda; validación en tiempo real con React Hook Form y Zod; los errores se expresan mediante texto rojo bajo el campo.
4. **Modal**: dialog accesible implementado con componentes de Radix; incluye aria-modal y gestión de foco; se usa para confirmaciones y carga de archivos.
5. **Toast/Alert**: notificaciones no obstructivas en esquina inferior; se cierran automáticamente y son anunciadas a lectores de pantalla mediante aria-live.

### Wireframes de referencia (ASCII)

#### Pantalla de acceso

┌──────────────────────────────┐  
│ Iniciar sesión │  
├──────────────────────────────┤  
│ Usuario [\_\_\_\_\_\_\_\_\_\_\_] │  
│ Clave [\_\_\_\_\_\_\_\_\_\_\_] │  
│ [Ingresar] │  
│ │  
│ ¿Olvidaste tu clave? │  
└──────────────────────────────┘

#### Grilla de cursos (administrador)

┌─────────────────────────────────────────────────────────────┐  
│ Cursos disponibles │  
├─────┬──────────┬──────────┬──────────┬────────────┬───────┤  
│ # │ Código │ Nombre │ Duración │ Modalidad │ Acciones │  
├─────┼──────────┼──────────┼──────────┼────────────┼─────────┤  
│ 1 │ C‑001 │ Excel │ 8 h │ Presencial │ Editar │  
│ │ │ │ │ │ Eliminar │  
│ 2 │ C‑002 │ Power BI │ 12 h │ Teams │ Editar │  
│ … │ … │ … │ … │ … │ … │  
└─────┴──────────┴──────────┴──────────┴────────────┴─────────┘  
│ [Nuevo curso] │  
└─────────────────────────────────────────────────────────────┘

#### Calendario de cursos (contratista)

┌───────────────────────────┐  
│ Septiembre 2025 │  
├───────────────────────────┤  
│ Lu Ma Mi Ju Vi Sa Do │  
│ 1 2 3 4 5 6 7 │ ← colores según tipo de curso  
│ 8 9 10 11 12 13 14 │ cada día muestra icono/tooltip  
│ 15 16 17 18 19 20 21 │  
│ 22 23 24 25 26 27 28 │  
│ 29 30 │  
└───────────────────────────┘  
│ Legend: │  
│ ● Azul = Propio │  
│ ● Gris = Otro │  
└───────────────────────────┘

#### Butacas y carga de nómina

┌─────────────────────────────────────────────┐  
│ Curso: Excel Avanzado – 10‑09‑2025 │  
├─────────────────────────────────────────────┤  
│ [Leyenda] Verde=Libre Rojo=Ocupado │  
│ │  
│ [●][●][○][○][○] … (200/30 asientos) │  
│ │  
│ Seleccionar butaca → Formulario manual │  
│ Cargar archivo Excel → Dialog de importación│  
│ │  
└─────────────────────────────────────────────┘

#### Asistencia y notas

┌─────────────────────────────────────────────────────────────┐  
│ Curso: Excel Avanzado (Fecha: 10‑09‑2025) │  
├─────┬───────────────┬───────────────┬─────────┬───────────┐  
│ # │ Nombre │ RUT │ Asistencia│ Nota │  
├─────┼───────────────┼───────────────┼─────────┼───────────┤  
│ 1 │ Juana Pérez │ 12.345.678‑5 │ 90 % │ 6.0 │  
│ 2 │ Pedro Gómez │ 98.765.432‑K │ 40 % │ 4.0 │ ← reprobado |  
│ … │ … │ … │ … │ … │  
└─────┴───────────────┴───────────────┴─────────┴───────────┘  
│ [Importar Excel] [Guardar cambios] [Exportar PDF] │  
└─────────────────────────────────────────────────────────────┘

#### Panel de usuario individual

┌──────────────────────────────┐  
│ Mis cursos │  
├──────────────────────────────┤  
│ Curso | Estado │  
│ Excel Avanzado | Aprobado │ [Descargar certificado]  
│ Power BI | Reprobado │ –  
│ … | … │ …  
└──────────────────────────────┘

# Diseño de Componentes (Front)

A continuación se describen los componentes principales por pantalla. Cada tabla incluye sus propiedades (Props), estado interno (Estado), eventos (Eventos), validaciones (Validaciones), mensajes de error (Errores) y consideraciones de accesibilidad (Accesibilidad).

## Pantalla de acceso

| Componente | Props | Estado | Eventos | Validaciones | Errores | Accesibilidad |
| --- | --- | --- | --- | --- | --- | --- |
| LoginForm | onSubmit (función), loading (bool) | usuario, clave, errors | handleChange, handleSubmit | Campos requeridos; longitud mínima de clave | Mensaje “Usuario/clave inválidos” | form con aria-labelledby; campos con label y aria-required; foco inicial en primer campo. |
| PasswordInput | value, onChange, label | showPassword | toggleVisibility | Ninguna (campo se valida en el formulario) | – | Incluye botón con aria-label para mostrar/ocultar. |
| SubmitButton | label, disabled | – | onClick | – | – | Rol button, estado disabled reflejado en atributo. |

## Gestión de cursos

| Componente | Props | Estado | Eventos | Validaciones | Errores | Accesibilidad |
| --- | --- | --- | --- | --- | --- | --- |
| CoursesTable | courses (array), onEdit, onDelete | sortBy, filter | handleSort, handleFilter, selectCourse | N/A (visualización) | – | Tablas accesibles con caption; cabeceras scope="col"; filas resaltables; permite navegación con teclado. |
| CourseFormModal | course (opcional), onSave, onClose | formData (nombre, código, fechas, etc.), errors | handleChange, handleSubmit, handleDateChange | Campos obligatorios; código único; formato de fecha | Mensajes bajo campo correspondiente | Modal con role="dialog", trampa de foco (se mantiene dentro); botones “Guardar” y “Cancelar” con etiquetas claras. |
| DeleteDialog | courseId, onConfirm, onCancel | – | handleConfirm, handleCancel | – | – | Confirmación accesible; se anuncia con aria-modal; permite cerrar con Esc. |

## Calendario y carga de nómina

| Componente | Props | Estado | Eventos | Validaciones | Errores | Accesibilidad |
| --- | --- | --- | --- | --- | --- | --- |
| CourseCalendar | courses (array), onSelectCourse | currentDate, selectedCourse | onDayClick, onEventClick, onNavigateMonth | Selección de fechas dentro del rango del curso | – | Usa FullCalendar con soporte de teclado; muestra título del mes y permite moverse con flechas y pageUp/PageDown【349603758647189†L301-L334】. |
| SeatMap | capacity, bookedSeats (array), onSeatSelect | selectedSeat | handleSeatClick, renderSeatStatus | Verificar que asiento esté libre antes de asignar | Mensaje “Butaca ocupada” | Uso de <button> para cada asiento con aria-label indicando posición; colores + texturas para no depender sólo de color. |
| ManualEnrollmentForm | onSubmit, onCancel, existingRuts (array) | name, rut, contractor, errors | handleChange, handleSubmit | Nombre requerido; RUT con módulo 11; no duplicado en existingRuts | Mensajes bajo campos; indicador de RUT inválido | form con descripción; en móvil se adapta; lectura de errores mediante aria-live. |
| BulkUploadDialog | onUpload, onCancel | file, preview, errors | handleFileSelect, handleImport, handleClose | Validar tipo de archivo (.xlsx); estructura de columnas; cantidad total ≤ capacidad | Lista de errores por fila (“RUT inválido”, “Nombre no coincide”) | Modal con foco inicial en input de archivo; instrucciones claras; permite cierre con botón y Esc. |

## Asistencia y calificaciones

| Componente | Props | Estado | Eventos | Validaciones | Errores | Accesibilidad |
| --- | --- | --- | --- | --- | --- | --- |
| AttendanceTable | participants (array), onImport, onEdit, onExport | sort, filters | handleSort, handleEdit, handleImport, handleExport | N/A (visualización) | – | Tabla accesible; permite ordenación via teclado; filas resaltadas. |
| ImportAttendanceDialog | onImport, onCancel | file, errors | handleFileSelect, handleImport | Validar columnas (RUT, asistencia, nota); tipos de dato; porcentajes entre 0 y 100 | Lista de filas con problemas. | Modal con instrucciones; roles adecuados; los errores se anuncian mediante aria-live. |
| ParticipantRowEditor | participant, onSave, onCancel | attendance, grade, errors | handleChange, handleSave | Asistencia entre 0 % y 100 %; nota entre 1,0 y 7,0; recalcular estado aprobado/reprobado | Mensajes por campo fuera de rango | Componente con role="row"; campos son input con etiquetas; se usa aria-describedby para errores. |

## Certificados y reportes

| Componente | Props | Estado | Eventos | Validaciones | Errores | Accesibilidad |
| --- | --- | --- | --- | --- | --- | --- |
| CertificateGenerator | course, participants | isGenerating | handleGenerate | Verificar que participantes estén aprobados | – | Botón con aria-label; progreso mostrado con barra e indicador; descarga se realiza mediante botón claro. |
| CertificateViewer | certificateData (base64) | – | handleDownload, handleClose | – | – | Modal con PDF embebido; permite zoom; se proporciona texto alternativo para contenido clave. |
| ReportsDashboard | data (agregada), filters | selectedCourse, dateRange, chartType | handleFilterChange, handleExport | Fechas válidas; rango no vacío | – | Gráficos incluyen descripciones, etiquetas y teclas de navegación; se utiliza aria-label en elementos interactivos【600224424098947†L122-L167】. |

# Servicios Simulados (Back/Domain)

La capa de dominio expone endpoints simulados que devuelven promesas para simular latencia. Se usan fetch wrappers que operan sobre IndexedDB. A continuación se listan los principales contratos en TypeScript/JSON.

## Endpoints y contratos

### Cursos

// GET /api/courses  
type Course = {  
 id: string;  
 codigo: string;  
 nombre: string;  
 duracionHoras: number;  
 fechas: { inicio: string; fin: string };  
 modalidad: 'presencial' | 'teams';  
 objetivos: string;  
};  
  
// POST /api/courses  
type CreateCourseRequest = Omit<Course, 'id'>;

### Sesiones y butacas

// GET /api/courses/:courseId/sessions  
type Session = {  
 id: string;  
 courseId: string;  
 fecha: string; // DD‑MM‑YYYY  
 capacity: number;  
 seats: Seat[];  
};  
  
type Seat = {  
 id: string;  
 sessionId: string;  
 estado: 'libre' | 'ocupado';  
 participantId?: string;  
};  
  
// POST /api/sessions/:sessionId/enrollments (inscripción manual o masiva)  
type EnrollmentRequest = {  
 sessionId: string;  
 participants: ParticipantRequest[];  
};  
  
type ParticipantRequest = {  
 nombre: string;  
 rut: string;  
 contractor: string;  
};

### Participantes, asistencia y calificaciones

// GET /api/participants?courseId=... devuelve todos los inscritos de un curso  
type Participant = {  
 id: string;  
 sessionId: string;  
 nombre: string;  
 rut: string;  
 contractor: string;  
 asistencia: number; // 0‑100  
 nota: number; // 1‑7  
 estado: 'aprobado' | 'reprobado';  
};  
  
// POST /api/attendance  
type AttendanceImportRequest = {  
 courseId: string;  
 rows: Array<{  
 rut: string;  
 asistencia: number;  
 nota: number;  
 }>;  
};

### Certificados y reportes

// POST /api/certificates/generate  
type CertificateGenerationRequest = {  
 courseId: string;  
};  
  
// GET /api/reports?start=DD‑MM‑YYYY&end=DD‑MM‑YYYY  
type ReportResponse = {  
 cursos: Array<{  
 courseId: string;  
 inscritos: number;  
 aprobados: number;  
 reprobados: number;  
 promedioAsistencia: number;  
 }>;  
};

## Orquestación de servicios

* **Creación/Edición de cursos**: guarda la entidad en IndexedDB y devuelve la lista actualizada.
* **Sesiones**: se generan automáticamente según las fechas del curso; la capacidad se define según modalidad (200 para Teams, 30 para presencial).
* **Inscripciones**: al recibir un EnrollmentRequest, el servicio verifica la capacidad restante, validez de RUT y duplicados antes de agregar Seat y Participant.
* **Importación de asistencia**: lee un archivo Excel mediante SheetJS; convierte filas a objetos y aplica reglas de negocio (porcentajes entre 0 y 100, notas entre 1 y 7).
* **Generación de certificados**: utiliza PDF‑Lib para componer un PDF por participante; se exporta en base64 y se envuelve en un ZIP simulado.
* **Reportes**: realiza consultas agregadas sobre IndexedDB (uso de Dexie) para devolver totales y promedios por curso y periodo.
* **Caché**: las consultas se memorizan en memoria para mejorar rendimiento de la demo.

# Modelo de Datos

## Diagrama Entidad‑Relación (texto)

[User] 1---\* [Enrollment] \*---1 [Session] \*---1 [Course]  
 | |  
 | \*---1 [Attendance]  
 |   
 \*---\* [Certificate]  
  
Entidades:  
- User: id (PK), nombre, rut, rol (Administrador/Contratista/Usuario), clave.  
- Course: id (PK), codigo, nombre, duracionHoras, fechaInicio, fechaFin, modalidad, objetivos.  
- Session: id (PK), courseId (FK), fecha, capacity.  
- Enrollment: id (PK), sessionId (FK), userId (FK), estado ('inscrito', 'aprobado', 'reprobado').  
- Attendance: id (PK), enrollmentId (FK), asistencia (0‑100), nota (1‑7).  
- Certificate: id (PK), enrollmentId (FK), pdfUrl (cadena).

## DDL SQL de referencia

CREATE TABLE User (  
 id TEXT PRIMARY KEY,  
 nombre TEXT NOT NULL,  
 rut TEXT UNIQUE NOT NULL,  
 rol TEXT NOT NULL CHECK (rol IN ('administrador', 'contratista', 'usuario')),  
 clave TEXT NOT NULL  
);  
  
CREATE TABLE Course (  
 id TEXT PRIMARY KEY,  
 codigo TEXT UNIQUE NOT NULL,  
 nombre TEXT NOT NULL,  
 duracion\_horas INTEGER NOT NULL,  
 fecha\_inicio DATE NOT NULL,  
 fecha\_fin DATE NOT NULL,  
 modalidad TEXT NOT NULL CHECK (modalidad IN ('presencial','teams')),  
 objetivos TEXT  
);  
  
CREATE TABLE Session (  
 id TEXT PRIMARY KEY,  
 course\_id TEXT NOT NULL REFERENCES Course(id),  
 fecha DATE NOT NULL,  
 capacity INTEGER NOT NULL  
);  
  
CREATE TABLE Enrollment (  
 id TEXT PRIMARY KEY,  
 session\_id TEXT NOT NULL REFERENCES Session(id),  
 user\_id TEXT NOT NULL REFERENCES User(id),  
 estado TEXT NOT NULL CHECK (estado IN ('inscrito','aprobado','reprobado'))  
);  
  
CREATE TABLE Attendance (  
 id TEXT PRIMARY KEY,  
 enrollment\_id TEXT NOT NULL REFERENCES Enrollment(id),  
 asistencia INTEGER NOT NULL CHECK (asistencia BETWEEN 0 AND 100),  
 nota REAL NOT NULL CHECK (nota BETWEEN 1 AND 7)  
);  
  
CREATE TABLE Certificate (  
 id TEXT PRIMARY KEY,  
 enrollment\_id TEXT NOT NULL REFERENCES Enrollment(id),  
 pdf\_url TEXT NOT NULL  
);

## Esquema JSON (mock)

{  
 "courses": [  
 {  
 "id": "uuid1",  
 "codigo": "C‑001",  
 "nombre": "Excel Avanzado",  
 "duracionHoras": 8,  
 "fechas": { "inicio": "10‑09‑2025", "fin": "12‑09‑2025" },  
 "modalidad": "presencial",  
 "objetivos": "Aprender funciones avanzadas."  
 }  
 ],  
 "sessions": [  
 {  
 "id": "uuid2",  
 "courseId": "uuid1",  
 "fecha": "10‑09‑2025",  
 "capacity": 30,  
 "seats": [  
 { "id": "s1", "sessionId": "uuid2", "estado": "libre" },  
 { "id": "s2", "sessionId": "uuid2", "estado": "ocupado", "participantId": "u3" }  
 ]  
 }  
 ],  
 "users": [  
 { "id": "u1", "nombre": "Admin", "rut": "1‑9", "rol": "administrador", "clave": "admin" },  
 { "id": "u2", "nombre": "Empresa X", "rut": "2‑7", "rol": "contratista", "clave": "1234" },  
 { "id": "u3", "nombre": "Juan Pérez", "rut": "12.345.678‑5", "rol": "usuario", "clave": "abcd" }  
 ],  
 "enrollments": [  
 { "id": "e1", "sessionId": "uuid2", "userId": "u3", "estado": "inscrito" }  
 ],  
 "attendance": [  
 { "id": "a1", "enrollmentId": "e1", "asistencia": 90, "nota": 6.0 }  
 ],  
 "certificates": [  
 { "id": "c1", "enrollmentId": "e1", "pdfUrl": "/certs/c1.pdf" }  
 ]  
}

# Validaciones y Reglas de Negocio

## Capacidad y duplicados

* **Capacidad**: Las sesiones derivan de la modalidad. Para Teams, la capacidad es de 200 participantes; para presencial, de 30. Al intentar inscribir más participantes, se rechaza la operación y se muestra un mensaje específico.
* **Duplicidad de participantes**: Antes de insertar un RUT en una sesión, se revisa la lista de inscripciones existentes. Si el RUT ya está inscrito en la misma sesión o en otra sesión del mismo curso, se genera una alerta.
* **Participación previa**: En cargas masivas se simula una verificación del historial; si se detecta que el participante ya aprobó el mismo curso en una fecha previa, se advierte para evitar redundancias.

## Validación de RUT chileno (módulo 11)

El número base del RUT se compone de 7 u 8 dígitos más un dígito verificador (DV). El cálculo se basa en el algoritmo Módulo 11【416033120628042†L189-L205】:

1. Eliminar puntos y guión del RUT. Separar el cuerpo (número) y el DV.
2. Asignar factores de chequeo desde 2 hasta 7 de derecha a izquierda, repitiéndolos cíclicamente【416033120628042†L189-L197】.
3. Multiplicar cada dígito por su factor y sumar los productos【416033120628042†L198-L201】.
4. Calcular el resto (suma mod 11) y restar el resultado a 11【416033120628042†L202-L206】.
5. Interpretar el resultado: si es 11, el DV es 0; si es 10, en Chile se usa la letra “K”; en otro caso el número mismo es el DV【416033120628042†L204-L233】.
6. Comparar el DV calculado con el DV ingresado; si coinciden, el RUT es válido.

### Pseudocódigo

function validarRUT(rut: string): { valido: boolean; mensaje: string } {  
 // paso 1: limpiar caracteres no numéricos  
 const limpio = rut.replace(/[^0-9Kk]/g, '').toUpperCase();  
 const cuerpo = limpio.slice(0, -1);  
 const dvIngresado = limpio.slice(-1);  
 if (cuerpo.length < 7) return { valido: false, mensaje: 'Formato insuficiente' };  
  
 // paso 2 y 3: aplicar factores  
 let suma = 0;  
 let factor = 2;  
 for (let i = cuerpo.length - 1; i >= 0; i--) {  
 suma += parseInt(cuerpo[i], 10) \* factor;  
 factor = factor === 7 ? 2 : factor + 1;  
 }  
 // paso 4: calcular resto y DV  
 const resto = suma % 11;  
 let dvCalculado = 11 - resto;  
 if (dvCalculado === 11) dvCalculado = 0;  
 else if (dvCalculado === 10) dvCalculado = 'K';  
 // paso 5: comparar  
 return {  
 valido: dvCalculado.toString() === dvIngresado,  
 mensaje: dvCalculado.toString() === dvIngresado ? 'Válido' : 'Dígito verificador incorrecto'  
 };  
}

## Regla de aprobación

* **Asistencia mínima**: Para aprobar un curso, el participante debe asistir al menos al 50 % de las sesiones (o 100 % si así lo define el curso).
* **Nota mínima**: En cursos con evaluación, se considera aprobado quien obtenga nota ≥ 4,0 (escala 1–7).
* **Estado final**: aprobado si cumple ambos criterios; reprobado en caso contrario. La maqueta recalcula el estado al modificar asistencia o nota.
* **Ejemplo**: un participante con 90 % de asistencia y nota 6,0 se marca como aprobado; otro con 40 % y nota 4,0 se marca reprobado.

## Casos borde

* RUT con DV “K” en minúscula (e.g., 98.765.432‑k) se normaliza a mayúscula antes de comparar.
* Inscripción masiva: si el Excel contiene filas vacías o repetidas, se omiten y se muestran en el resumen de errores.
* Si se cancelan inscripciones después de generar certificados, los certificados generados quedarán disponibles hasta que el administrador los elimine manualmente.

# Tecnologías y Frameworks (2025)

El objetivo de la maqueta es demostrar interactividad y mejores prácticas. A continuación se detallan las tecnologías recomendadas con justificación y alternativas.

| Categoría | Recomendación principal | Justificación | Alternativas | Pros/Contras de alternativas |
| --- | --- | --- | --- | --- |
| **Framework** | **Next.js 14 con TypeScript** | Permite desarrollo rápido, rutas flexibles y capacidad de migrar a SSR si se requiere en el futuro; soporta carga de imágenes y archivos de forma nativa y tiene comunidad extensa. | Vue 3 con Vite; SvelteKit | Vue: sintaxis más simple pero menos ecosistema; SvelteKit: excelente performance y simplicidad pero comunidad más pequeña y menos librerías disponibles. |
| **Estilos/UI** | **Tailwind CSS + shadcn/ui** | Sistema de utilidades que acelera el prototipado; tokens configurables y buen soporte de accesibilidad; shadcn/ui ofrece componentes accesibles basados en Radix. | Chakra UI; Material UI | Chakra: componentes accesibles pero con sobrecarga CSS; Material UI: completo pero el look and feel puede no alinearse con branding propio. |
| **Estado** | **Zustand** | API minimalista para manejar estado global sin boilerplate; fácil de combinar con React Suspense y persistencia; adecuado para maquetas. | Redux Toolkit; Pinia | Redux Toolkit: robusto pero verboso; Pinia: específico de Vue. |
| **Formularios** | **React Hook Form + Zod** | Gestión de formularios desacoplada del DOM; validación basada en esquemas con Zod; integración con TypeScript. | Formik + Yup; VeeValidate | Formik: mayor tamaño y menos performance; VeeValidate: específico de Vue. |
| **Calendario** | **FullCalendar React** | Soporte completo de vistas, eventos recurrentes y personalización; accesibilidad a través de ARIA y navegación por teclado. | react-big-calendar | Menos pesado, pero con opciones de personalización limitadas y accesibilidad menos documentada. |
| **Tablas** | **TanStack Table** | Librería headless que permite total control de la presentación y cumple con normas de accesibilidad; fácilmente integrable con React. | AG Grid; DataTables | AG Grid: potente pero con licencia y mayor complejidad; DataTables: requiere jQuery y no está optimizado para React. |
| **Excel** | **SheetJS (xlsx)** | Lectura y escritura de archivos Excel en el navegador; soporta grandes datasets; configurable para lectura densificada (dense mode)【224872993310552†L68-L100】. | PapaParse (CSV); ExcelJS | PapaParse: sólo CSV, no XLSX; ExcelJS: generación de Excel pero lectura menos robusta y mayor peso. |
| **PDF** | **PDF‑Lib** | Permite crear y modificar PDFs en el navegador; soporte de fuentes y textos complejos; liviano en comparación con otras librerías. | jsPDF | jsPDF es más simple pero ofrece menor control sobre estilos y objetos complejos. |
| **Gráficos** | **Recharts** | Componentes React basados en D3; admiten animaciones, personalización y permiten incorporación de descripciones y etiquetas para accesibilidad【600224424098947†L122-L167】. | Chart.js + react-chartjs-2 | Chart.js es popular pero requiere wrapper y no integra accesibilidad por defecto. |
| **Persistencia** | **IndexedDB + Dexie** | Base de datos local asíncrona; permite almacenar objetos estructurados; Dexie simplifica consultas y transacciones; adecuada para datos temporales. | localStorage; PouchDB | localStorage: rápido pero limitado y sin transacciones; PouchDB: potente y replicable pero sobredimensionado para una maqueta. |
| **Testing** | **Playwright + Vitest** | Playwright permite pruebas de extremo a extremo con múltiples navegadores y soporte de accesibilidad; Vitest es ligero y compatible con TypeScript. | Cypress; Jest | Cypress: popular pero sólo Chromium y Firefox; Jest: robusto para unit testing pero requiere configuración extra para E2E. |

# Plan y Calendario (2 semanas)

El proyecto se desarrolla entre el lunes **01‑09‑2025** y el domingo **14‑09‑2025**. Se trabaja de lunes a viernes, reservando los fines de semana para ajustes menores. Cada actividad tiene un entregable y un criterio de aceptación que permitirá medir el avance.

| Fecha | Actividad | Entregable | Responsable | Criterio de aceptación |
| --- | --- | --- | --- | --- |
| 01‑09‑2025 | Kickoff, desglosar requisitos y elaborar IA | Lista de REQ con identificadores; mapa de IA | Equipo completo | Todos los requisitos del archivo base identificados y trazados en el documento. |
| 02‑09‑2025 | Definir flujos y crear wireframes de Acceso y Cursos | Wireframes iniciales (acceso, grilla de cursos) | UX Lead | Flujos navegables dibujados; consenso con stakeholders. |
| 03‑09‑2025 | Diseñar calendario y validar reglas de capacidad | Prototipo de calendario con selección de día | Front Lead | Calendario reactivo con navegación por teclado; validación de capacidad (30/200) operativa. |
| 04‑09‑2025 | Implementar carga masiva con SheetJS | Demo de importación de Excel | Front + Domain | Lectura de archivo Excel; detección de errores por fila; resumen de filas cargadas vs. fallidas. |
| 05‑09‑2025 | Desarrollar pantalla de asistencia y notas | Pantalla de asistencia con cálculo de estados | Front Lead | Importación de planilla de asistencia; cálculo de % y nota; estado aprobado/reprobado recalculado. |
| 06‑09‑2025 | Ajustes y revisión de UX | Ajustes menores en wireframes y estilos | UX Lead | Comentarios de usuarios internos incorporados; consistencia visual y accesibilidad. |
| 08‑09‑2025 | Generar certificados PDF | Prototipo de certificados en PDF | Domain Lead | Creación de PDF por participante; descarga individual y masiva (ZIP simulado) sin errores de formato. |
| 09‑09‑2025 | Implementar dashboard de reportes y gráficas | Dashboard con gráficos y filtros | Front + Data | Gráficos interactivos con filtros; exportación a PDF; datos correctos en totales y promedios. |
| 10‑09‑2025 | Asegurar accesibilidad y responsive | Checklist de WCAG 2.2 AA | QA Lead | Cumplimiento de criterios: foco visible, tamaño de objetivos (24×24 px), ayuda consistente, etc【503959695386811†L246-L284】. |
| 11‑09‑2025 | Pruebas E2E y estabilización | Evidencias de pruebas | QA Lead | Ejecución de pruebas Playwright en flujos: inscripción manual/masiva, asistencia, certificados y reportes. |
| 12‑09‑2025 | Preparar demo y documentación final | Maqueta navegable + documentación | Equipo completo | Demo funcional sin errores críticos; documento técnico‑funcional revisado y corregido. |
| 13‑09‑2025 | Revisión por stakeholders | Feedback y lista de ajustes finales | Stakeholders | Aprobación preliminar; identificación de mejoras para versión posterior. |
| 14‑09‑2025 | Ajustes finales y cierre | Maqueta final + entrega oficial | Equipo completo | Se cumplen todos los criterios de aceptación; entrega publicada en Vercel con README incluido. |

# QA, Accesibilidad y Performance

## Checklist WCAG 2.2 AA (resumen)

* **Textos alternativos**: todas las imágenes y elementos no textuales disponen de descripciones.
* **Contraste**: colores cumplen ratio mínimo de 4.5:1; se verifica en botones, textos y iconos.
* **Foco visible**: los componentes muestran un indicador de foco claro; no queda oculto por otros elementos según criterios 2.4.11 y 2.4.12【503959695386811†L246-L253】.
* **Tamaño objetivo**: las áreas interactivas miden al menos 24×24 px según SC 2.5.8【503959695386811†L264-L266】.
* **Navegación con teclado**: todas las funciones (calendario, tablas, modales) se pueden operar con teclado sin atraparse; se respeta orden lógico de tabulación.
* **Ayuda consistente**: enlaces a documentación y soporte siempre visibles en la misma ubicación (criterio 3.2.6)【503959695386811†L268-L274】.
* **Autenticación accesible**: la pantalla de login permite pegar contraseña y utilizar gestores (criterio 3.3.8)【503959695386811†L277-L283】.
* **Redundancia de entrada**: la plataforma evita pedir datos repetidos (criterio 3.3.7)【503959695386811†L272-L275】.
* **Arrastre alternativo**: el calendario permite navegación por clics además de arrastrar (criterio 2.5.7)【503959695386811†L259-L263】.

## Pruebas clave

1. **Flujo de inscripción manual**: iniciar sesión como contratista, seleccionar curso y día, inscribir participantes manualmente; verificar validaciones y estado de butacas.
2. **Flujo de carga masiva**: cargar archivo Excel con casos válidos, RUT inválidos y duplicados; comprobar detección de errores y resumen.
3. **Flujo de asistencia y notas**: importar planilla de asistencia y notas; revisar cálculo de porcentajes y estados; editar valores y confirmar recalculación.
4. **Flujo de certificados**: generar certificados masivos e individuales; abrir PDFs y comprobar legibilidad y datos correctos.
5. **Flujo de reportes**: navegar por dashboard, aplicar filtros y exportar reportes; verificar que los datos coincidan con la base simulada.
6. **Accesibilidad**: usar lector de pantalla (NVDA o similar) para navegar; validar que cada componente anuncie su propósito; probar navegación con teclado y uso de focos.
7. **Performance**: medir tiempos de carga inicial (<3 s) y respuesta de componentes; simular carga con 200 participantes.

## Performance

* Se aplican estrategias de carga diferida (lazy loading) para módulos pesados como reportes y generación de PDFs.
* Uso de useMemo y useCallback para evitar renders innecesarios; estado centralizado en hooks de Zustand.
* IndexedDB opera de forma asíncrona, evitando bloqueos del hilo principal.
* Uso de dense mode en SheetJS para mejorar rendimiento al procesar hojas grandes【224872993310552†L68-L100】.
* Los gráficos se renderizan únicamente cuando los datos están disponibles y se desmontan al salir de la vista.

# Riesgos y Mitigaciones

| Riesgo | Impacto | Probabilidad | Mitigación |
| --- | --- | --- | --- |
| Complejidad en validaciones de RUT y capacidad | Errores en inscripción, mala UX | Media | Implementar funciones unitarias para cada validación; realizar pruebas unitarias y E2E; mensajes claros. |
| Rendimiento con listas grandes (200 participantes) | UI lenta y experiencia frustrante | Media | Utilizar paginación virtualizada en tablas; activar dense mode en SheetJS; optimizar renders con memoización. |
| Inconsistencia visual entre módulos | Percepción de poca calidad | Baja | Definir design tokens y componentes reutilizables; revisión constante de UX Lead. |
| Falta de adopción del calendario/tabla accesible | Usuarios con discapacidad no pueden usar la maqueta | Baja | Seguir guías de WAI para tablas y calendarios【484670439075094†L94-L146】【349603758647189†L301-L334】; pruebas con teclado y lector de pantalla. |
| Exceso de alcance en 2 semanas | Retraso en entrega final | Media | Priorizar funcionalidades críticas; mantener backlog de mejoras para iteraciones futuras. |

# Matriz de Trazabilidad

La siguiente matriz relaciona cada requisito con las pantallas, entidades de datos, validaciones y pruebas asociadas. Esta trazabilidad asegura que todos los elementos se tengan en cuenta durante el desarrollo y la prueba.

| REQ | UI/Pantalla | Entidad(es) | Validaciones/Reglas | Pruebas (QA) |
| --- | --- | --- | --- | --- |
| A1 | Acceso | User | Campos obligatorios | Prueba 1 (login correcto/incorrecto) |
| A2 | Cursos (grilla, formulario) | Course | Código único, formato fecha | Prueba 2 (crear/editar/eliminar curso) |
| A3 | Cursos | Course | Confirmación al eliminar | Prueba 2 |
| A4 | Asistencia (importar Excel) | Attendance, Enrollment | Formato Excel, % asistencia | Prueba 3 (importación y cálculo) |
| A5 | Asistencia (editor de filas) | Attendance | Rangos de asistencia y nota | Prueba 3 |
| A6 | Certificados | Certificate | Participantes aprobados | Prueba 4 |
| A7 | Reportes | ReportResponse | Fechas válidas, filtros | Prueba 5 |
| C1 | Acceso | User | Campo de clave requerido | Prueba 1 |
| C2 | Calendario de cursos | Session, Course | Colores por tipo | Prueba 1, 2 |
| C3 | Calendario/Butacas | Session, Seat | Capacidad (30/200) | Prueba 1 |
| C4 | Carga manual de nómina | Participant, Seat | RUT válido, no duplicado | Prueba 1 |
| C5 | Carga masiva de nómina | Participant, Seat | Estructura de Excel, capacidad | Prueba 2 |
| C6 | Validaciones y alertas | Participant | Capacidad, RUT, duplicados | Pruebas 1 y 2 |
| U1 | Panel de usuario, descarga de certificados | Certificate, Enrollment | Estado aprobado | Prueba 4 |

# Anexos

## Snippet TypeScript para persistencia en IndexedDB (Dexie)

import Dexie, { Table } from 'dexie';  
  
export interface Course {  
 id: string;  
 codigo: string;  
 nombre: string;  
 duracionHoras: number;  
 fechaInicio: string;  
 fechaFin: string;  
 modalidad: 'presencial' | 'teams';  
 objetivos: string;  
}  
  
export class AppDB extends Dexie {  
 courses!: Table<Course, string>;  
 constructor() {  
 super('appDB');  
 this.version(1).stores({  
 courses: 'id,codigo,nombre',  
 sessions: 'id,courseId,fecha',  
 enrollments: 'id,sessionId,userId',  
 attendance: 'id,enrollmentId',  
 certificates: 'id,enrollmentId'  
 });  
 }  
}  
  
export const db = new AppDB();  
  
// Ejemplo de inserción  
export async function addCourse(course: Course) {  
 await db.courses.add(course);  
}

## Ejemplo de importación con SheetJS y validación básica

import \* as XLSX from 'xlsx';  
  
export async function parseEnrollmentFile(file: File, capacity: number) {  
 const data = await file.arrayBuffer();  
 const wb = XLSX.read(data, { type: 'array', dense: true });  
 const sheet = wb.Sheets[wb.SheetNames[0]];  
 const rows: any[] = XLSX.utils.sheet\_to\_json(sheet, { header: 0 });  
 const errors: string[] = [];  
 const participants: ParticipantRequest[] = [];  
 rows.forEach((row, index) => {  
 const nombre = row['Nombre'];  
 const rut = row['RUT'];  
 const contractor = row['Contratista'];  
 const { valido, mensaje } = validarRUT(rut);  
 if (!nombre || !rut || !contractor) {  
 errors.push(`Fila ${index + 2}: campos incompletos`);  
 return;  
 }  
 if (!valido) {  
 errors.push(`Fila ${index + 2}: RUT inválido (${mensaje})`);  
 return;  
 }  
 participants.push({ nombre, rut, contractor });  
 });  
 if (participants.length > capacity) {  
 errors.push('La nómina supera la capacidad del curso');  
 }  
 return { participants, errors };  
}

## Ejemplo de generación de certificado con PDF‑Lib

import { PDFDocument, StandardFonts, rgb } from 'pdf-lib';  
  
export async function generarCertificado(nombre: string, curso: string, fecha: string) {  
 const pdfDoc = await PDFDocument.create();  
 const page = pdfDoc.addPage([595.28, 841.89]); // A4 vertical  
 const font = await pdfDoc.embedFont(StandardFonts.Helvetica);  
 page.drawText('Certificado de Aprobación', { x: 50, y: 780, size: 24, font, color: rgb(0, 0.2, 0.5) });  
 page.drawText(`Se certifica que`, { x: 50, y: 720, size: 12, font });  
 page.drawText(`${nombre}`, { x: 50, y: 700, size: 18, font, color: rgb(0.1, 0.1, 0.1) });  
 page.drawText(`ha aprobado el curso`, { x: 50, y: 680, size: 12, font });  
 page.drawText(`${curso}`, { x: 50, y: 660, size: 16, font, color: rgb(0.1, 0.1, 0.1) });  
 page.drawText(`Fecha: ${fecha}`, { x: 50, y: 640, size: 12, font });  
 page.drawText('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_', { x: 50, y: 580, size: 12, font });  
 page.drawText('Firma digital', { x: 50, y: 560, size: 10, font });  
 const pdfBytes = await pdfDoc.save();  
 return pdfBytes;  
}

Este documento integra análisis, diseño y plan de implementación para una maqueta interactiva de gestión de cursos. La estructura propuesta permite avanzar de manera ordenada, trazando cada requisito hasta su prueba, y facilita la evolución hacia un producto funcional en iteraciones posteriores.