UNIVERSIDAD AUTÓNOMA DE YUCATÁN

FACULTAD DE MATEMÁTICAS

CONSTRUCCIÓN DE SOFTWARE

EQUIPO # 5

INTEGRANTES

EMILIANO CONTRERAS GAMBOA

HUGO JANSSEN AGUILAR

MARCOS OSORIO RODRIGUES PIÑA

GRECO ALEJANDRO GACHUZ PIÑA

Documento de descripción de requisitos

1. Descripción del proyecto

Aplicación web Cliente-Servidor diseñada para la generación dinámica de documentos PDF. El propósito principal es permitir a los usuarios generar PDFs fusionando plantillas con datos específicos. Los usuarios podrán utilizar un catálogo de plantillas predefinidas por el sistema o, alternativamente, crear, gestionar y almacenar sus propias plantillas personalizadas.

El núcleo del backend será una API RESTful desarrollada con el framework Spring Boot. Esta API manejará toda la lógica de negocio, incluyendo la autenticación de usuarios, la gestión de plantillas (CRUD) y el procesamiento de archivos. En lugar de un patrón MVC tradicional que renderiza HTML, el backend expondrá endpoints que recibirán peticiones y responderán con datos en formato JSON. El cliente será una Aplicación de Página Única (SPA) estática, compuesta por index.html, style.css y script.js, cuya única responsabilidad será consumir la API REST y renderizar la interfaz de usuario en el navegador.

El sistema contará con un módulo de gestión de usuarios, permitiendo el registro e inicio de sesión. La autenticación se manejará mediante tokens JWT para asegurar las peticiones a la API. Se definirán roles, como USUARIO (que gestiona sus plantillas privadas) y ADMIN (que gestiona las plantillas públicas del sistema).

las plantillas HTML deberán contener placeholders (marcadores de posición, ej. {{nombre_cliente}}). El usuario enviará los datos dinámicos correspondientes en formato JSON a un endpoint específico de la API. El backend validará esta información, fusionará la plantilla HTML/CSS con los datos JSON proporcionados y generará el archivo PDF resultante, el cual será devuelto al cliente para su descarga inmediata.

2. Requisitos funcionales

RF-01 Interfaz de Usuario para Entrada de Datos: El sistema debe proveer una interfaz web simple (formulario HTML) donde el usuario pueda introducir los datos necesarios para generar un documento.

RF-02 Selección de Múltiples Plantillas: El usuario debe poder elegir entre varias plantillas de documentos disponibles (ej. "Cotización", "Factura", "Reporte de Actividad") a través de un menú desplegable en la interfaz.

RF-03 Envío de Datos al Servidor: Al enviar el formulario, el cliente web debe empaquetar los datos del usuario y el tipo de plantilla seleccionada en un formato **JSON** y enviarlos al servidor mediante una petición HTTP POST.

RF-04 Generación de Documentos PDF: El servidor debe ser capaz de recibir la petición, interpretar los datos del JSON y utilizar la plantilla correspondiente para generar un documento PDF con la información incrustada.

RF-05 Descarga del Documento Generado: Una vez generado el PDF, el servidor debe devolverlo al cliente de tal forma que el navegador del usuario inicie automáticamente la descarga del archivo.

RF-06 Formulario Dinámico: La interfaz de usuario debe **adaptarse dinámicamente** según la plantilla seleccionada. Si el usuario elige "Factura", el formulario debe mostrar campos para "Número de Factura" y "Items". Si elige "Certificado", debe mostrar campos para "Nombre del Galardonado" y "Nombre del Curso". Esto evita mostrar campos irrelevantes.

RF-07 Feedback Visual para el Usuario: El sistema debe proporcionar feedback visual al usuario durante el proceso. Al hacer clic en "Generar PDF", el botón debería desactivarse y mostrar un estado de "Cargando..." o un spinner. Después de la respuesta del servidor, el sistema debe mostrar un mensaje claro de "Éxito" o "Error" si la generación del PDF falló.

RF-08: Validación de Datos en el Servidor: El servidor debe validar los datos recibidos en el JSON antes de intentar generar el PDF. Si faltan campos obligatorios o los datos no tienen el formato correcto (ej. texto en un campo numérico), el servidor debe rechazar la petición y devolver un mensaje de error claro al cliente. Esto previene que el servidor falle por datos incorrectos.

3. **Requisitos no funcionales**

RNF-01 Mantenibilidad y Extensibilidad: La arquitectura del servidor será modular y fuertemente desacoplada (aplicando el patrón MVC). Deberá ser posible añadir soporte para una nueva plantilla de documento con modificaciones mínimas y aisladas, sin impactar la funcionalidad existente. El código estará documentado para facilitar su comprensión.

RNF-02 Rendimiento: El tiempo de procesamiento en el servidor, desde que recibe una petición válida hasta que inicia el envío del PDF, no deberá exceder los 3 segundos en condiciones de carga normal.

RNF-03 Robustez y Manejo de Errores: El servidor debe ser resiliente a entradas incorrectas. En caso de recibir un JSON malformado o con datos incompletos, la aplicación no debe fallar. En su lugar, deberá responder de forma controlada con un código de estado HTTP de error (ej. 400 Bad Request) y un mensaje descriptivo.

RNF-04 Independencia y Portabilidad: El servidor será una aplicación autocontenida, ejecutable en cualquier sistema operativo con una Java Virtual Machine (JVM) compatible, sin requerir un servidor de aplicaciones externo (como Tomcat). Su huella de dependencias será mínima para garantizar la simplicidad y portabilidad.

RNF-05 Compatibilidad del Cliente: La interfaz de usuario debe funcionar correctamente en las últimas dos versiones estables de los navegadores web modernos (Google Chrome, Mozilla Firefox, Microsoft Edge).

4. Alcance y Limitaciones del Proyecto

Dentro del Alcance:

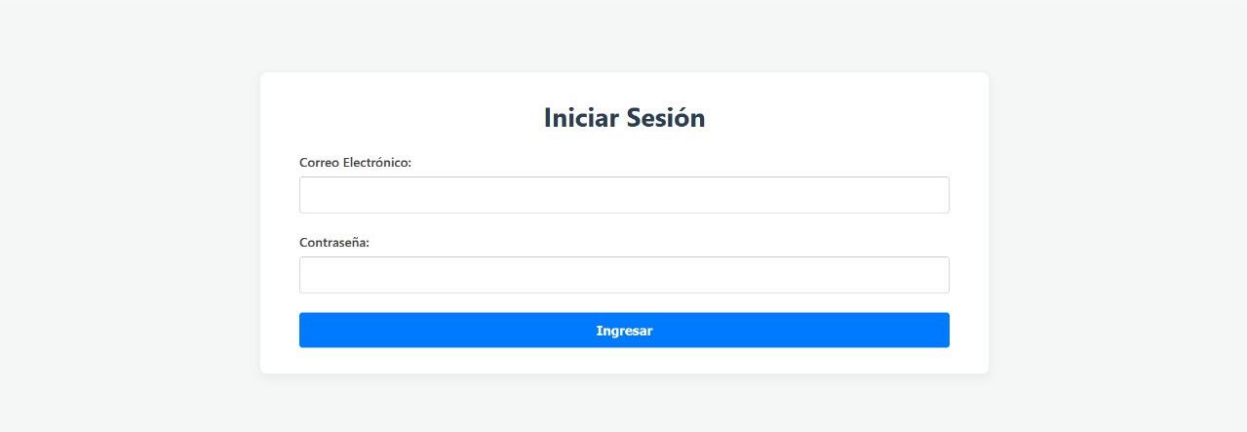
- Todo lo descrito en los Requisitos Funcionales.
- La creación de 6 a 8 plantillas PDF de ejemplo para demostrar la funcionalidad.
- Una estructura de código limpia y comentada en el servidor Java.

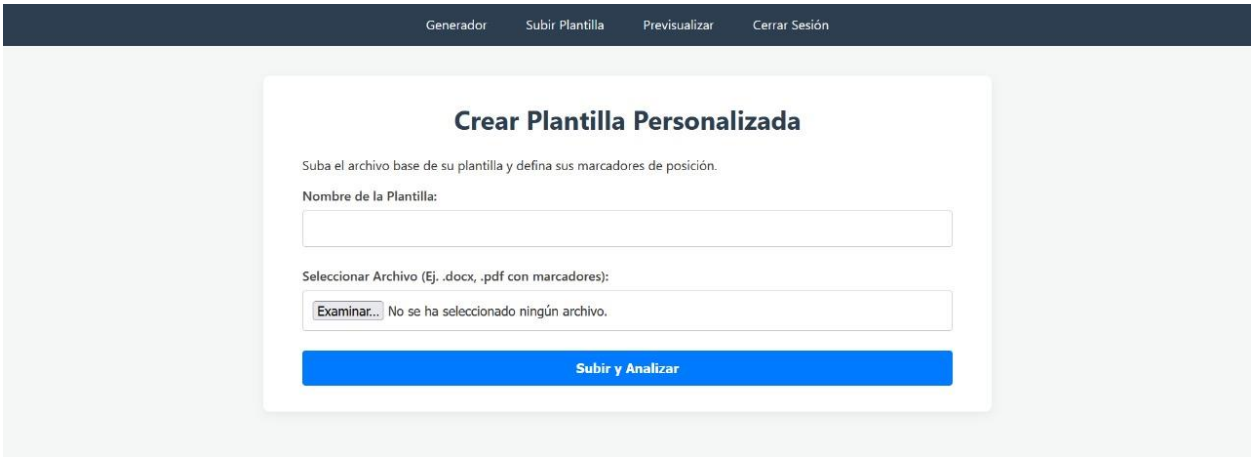
Fuera del Alcance (Limitaciones):

- **No habrá persistencia de datos:** El sistema no guardará en una base de datos los reportes generados ni los datos enviados. Cada petición es independiente.
- **No habrá autenticación de usuarios:** La aplicación será de acceso público sin necesidad de login.
- **No habrá un editor de plantillas:** Las plantillas PDF se crearán manualmente con herramientas externas y se añadirán al proyecto; el sistema permite la creación de plantillas
- **El cliente web tendrá una funcionalidad mínima:** No se invertirá tiempo en un diseño complejo ni en validaciones avanzadas en el frontend. Su propósito es ser funcional.

5. Mockup

A continuación, se presenta un Mockup de baja fidelidad para demostrar a grandes razgos como se verá el producto del proyecto.





6. Estandar de programación

El estandar de programación que se va a utilizar a lo largo del proyecto es el de la Convención de Código de Java de Oracle.

Referencia: https://www.oracle.com/docs/tech/java/codeconventions.pdf

7. Plan de Trabajo.

Fase	Tarea	Responsables	Fechas	Requisitos
Fase 1: MVP	**Configuración del Proyecto Backend (Spring Boot)** Configurar el proyecto, dependencias (web, pdf), y estructura de paquetes.	**Marcos, Hugo**	5 Nov. - 6 Nov.	RNF-01, RNF-04
Fase 1: MVP	**Servicio de Generación de PDF (Lógica)** Implementar la lógica para tomar un JSON y fusionarlo con una plantilla HTML/CSS *hardcodeada* para generar un PDF. Incluir prueba unitaria.	**Marcos, Emiliano**	6 Nov. - 7 Nov.	RF-04
Fase 1: MVP	**Endpoint de Generación (API)** Crear un endpoint POST (ej. /api/generate) que reciba el JSON, llame al servicio de Marcos/Emiliano y devuelva el PDF.	**Hugo, Greco**	7 Nov. - 8 Nov.	RF-03, RF-05
Fase 1: MVP	**Estructura Base del Frontend (HTML/CSS)** Crear la estructura (index.html, style.css, script.js) y un formulario HTML *estático*	**Emiliano, Greco**	6 Nov. - 7 Nov.	RF-01

	simple (basado en la plantilla *hardcodeada*).			
Fase 1: MVP	**Lógica Frontend (JS)** Escribir la lógica JS (fetch) para tomar los datos del formulario, crear el JSON, enviarlo al endpoint y forzar la descarga del PDF.	**Emiliano, Greco**	8 Nov. - 9 Nov.	RF-03, RF-05
Fase 1: MVP	**Integración y Pruebas del MVP** Conectar y probar el flujo completo. Resolver bugs. Preparar la demo.	**Todos**	10 Nov. - 11 Nov.	-
Fase 2: Autenticación y Plantillas	**Módulo de Usuarios (Backend)** Implementar Modelo, Repositorio, Servicio para Usuarios. Incluir endpoints de Registro (/register) e Inicio de Sesión (/login).	**Hugo, Marcos**	12 Nov. - 15 Nov.	-
Fase 2: Autenticación y Plantillas	**Implementación de Seguridad (Backend)** Implementar autenticación con JWT (generación, validación) y Spring Security.	**Hugo, Greco**	14 Nov. - 17 Nov.	-

Fase 2: Autenticación y Plantillas	**Módulo de Plantillas (Backend)** Implementar Modelo, Repositorio, Servicio para Plantillas. Crear endpoints CRUD (POST, GET, DELETE) y asegurarlos con JWT.	**Marcos, Emiliano**	15 Nov. - 18 Nov.	-
Fase 2: Autenticación y Plantillas	**Pruebas Unitarias (Backend)** Escribir pruebas unitarias para UserService, TemplateService y AuthController.	**Hugo, Marcos**	18 Nov. - 19 Nov.	-
Fase 2: Autenticación y Plantillas	**Vistas de Autenticación (Frontend)** Crear las vistas de "Iniciar Sesión" y "Registro". Escribir lógica JS para llamar a los endpoints de auth, almacenar JWT y proteger vistas.	**Greco, Emiliano**	16 Nov. - 19 Nov.	Mockup
Fase 2: Autenticación y Plantillas	**Vistas de Gestión de Plantillas (Frontend)** Crear vista "Mis Plantillas" (listar) y "Subir Plantilla" (formulario). Conectar estas vistas a los endpoints CRUD del backend.	**Emiliano, Marcos**	19 Nov. - 22 Nov.	Mockup

Fase 3: Integración Final	**Formulario Dinámico (Frontend)** **Tarea Clave:** Lógica JS que: 1. Lea plantillas. 2. Al seleccionar una, analice sus placeholders (ej. {{nombre}}). 3. Genere dinámicamente el formulario.	**Emiliano, Hugo**	23 Nov. - 26 Nov.	RF-06
Fase 3: Integración Final	**Feedback Visual (Frontend)** Añadir estados de "Cargando..." (spinners) a los botones y mostrar mensajes claros de "Éxito" o "Error" (RF-07).	**Greco, Marcos**	25 Nov. - 27 Nov.	RF-07
Fase 3: Integración Final	**Validación y Errores (Backend)** Implementar Validación de Datos en el Servidor (RF-08). Asegurar que la API devuelva errores 400 con mensajes claros (RNF-03).	**Hugo, Greco**	26 Nov. - 28 Nov.	RF-08, RNF-03
Fase 3: Integración Final	**Pruebas de Integración y Rendimiento** Liderar la creación de Pruebas de Integración de API (probando endpoints protegidos). Asegurar que el	**Marcos, Hugo**	28 Nov. - 30 Nov.	RNF-02

	rendimiento cumpla con el RNF-02 (< 3 seg).			
Fase 3: Integración Final	**Pruebas Finales y Documentación** Pruebas de extremo a extremo (manuales). Escribir/Completar la documentación (README.md). Preparar la presentación final.	**Todos**	1 Dic. - 3 Dic.	RNF-01, RNF-05