PROYECTO

ACTIVIDADES EXTRACURRICULARES

DOCUMENTACIÓN

PRESENTA

Castillo Navarrete Bryan Iván

Rubío Hernández Jonathan Jared

Alonso Jiménez Josué

Galindo Morales Héctor

Gutiérrez Lara Carlos Osiel

NOMBRE DEL PROFESOR(A): LORENA RAMÍREZ CAMPOY

FECHA DE ENTREGA DEL TRABAJO: 31/10/2025

PROPUESTA DE PROYECTO

En la Universidad Tecnológica de Tecámac, los estudiantes tienen dificultades para encontrar y registrarse en actividades extracurriculares (talleres, cursos, deportes, eventos culturales).

Actualmente la información está dispersa en carteles, publicaciones en redes Sociales y grupos de WhatsApp, lo que provoca: baja participación en actividades, sobrecupo en algunos eventos y baja asistencia en otros, falta de estadísticas para la coordinación escolar.

Propuesta de solución

Desarrollar una aplicación web, que centralice toda la información de actividades extracurriculares, permita registro en línea y genere reportes para la gestión.

Objetivo general:

Desarrollar una aplicación web para la Universidad Tecnológica de Tecámac que centralice la información de actividades extracurriculares, permita a los estudiantes registrarse en línea y proporcione a la coordinación escolar herramientas de gestión y generación de reportes, con el fin de optimizar la organización, incrementar la participación estudiantil y mejorar la toma de decisiones.

Objetivos específicos:

- Analizar los requerimientos de estudiantes y coordinadores de las actividades para definir las funcionalidades principales del sistema.
- Diseñar una arquitectura cliente-servidor basada en API REST que permita escalabilidad y facilite futuras integraciones móviles.
- Implementar un login y registro de usuarios que gestionen correctamente los integrantes del equipo (admin).
- Desarrollar un módulo de eventos con operaciones CRUD (crear, leer, actualizar, eliminar) y control de cupo por actividad.
- Construir una interfaz web intuitiva y responsiva con CSS que facilite la experiencia de los usuarios (estudiantes).
- Incorporar un sistema de notificaciones (correo o alertas web) para confirmar inscripciones y mantener informados a los usuarios.
- Generar reportes y estadísticas sobre participación estudiantil, accesibles en un panel administrativo.
- Realizar pruebas funcionales, de seguridad y de usabilidad para garantizar la calidad del sistema.

- Documentar el proceso de desarrollo, la arquitectura utilizada y el manual de usuario para asegurar la correcta adopción de la herramienta.

Metodología de desarrollo

SCRUM

La metodología SCRUM es la más adecuada para este proyecto, ya que permite iteraciones cortas y retroalimentación constante de los usuarios finales (estudiantes y administración de la universidad). Esto asegura que el producto evolucione de acuerdo a las necesidades reales y no se quede solo en la planeación inicial.

Características principales

- Se disminuye el tiempo de desarrollo al dividir el proyecto en funcionalidades pequeñas e incrementales.
- Se realizan sprints de 3 días, donde en cada uno se avanza en funcionalidades específicas y se entrega un incremento funcional en GitHub.
- Se mantiene un control de versiones en GitHub, todo en la rama principal.
- Se garantiza la retroalimentación constante de alumnos (usuarios finales).

Roles dentro del equipo

- **Líder del equipo:** Responsable de facilitar el proceso, eliminar impedimentos y asegurar que se cumpla SCRUM.
- **Equipo de desarrollo:**
    - **Programadores:** Frontend en Laravel / Backend en Node.js.
    - **Analista de datos:** Estructura de BD, reportes y métricas.

SCRUM se adapta perfectamente porque:

1. El proyecto requiere validación continua de usuarios (estudiantes y admins).
2. Las funcionalidades pueden dividirse en módulos pequeños (registro, eventos, inscripciones, reportes).
3. Permite flexibilidad: si cambia un requerimiento (ej. agregar códigos QR a inscripciones), se ajusta en el backlog sin rehacer todo el sistema.
4. La entrega incremental asegura que el sistema esté funcionando aún antes de finalizar las 13 semanas.

Arquitectura

CLIENTE-SERVIDOR CON NODE JS

La aplicación web se desarrollará bajo una arquitectura Cliente–Servidor, donde el cliente será el frontend encargado de la interacción con los estudiantes y administradores, y el servidor será el backend con Node.js, responsable de procesar la lógica de negocio, gestionar solicitudes y comunicarse con la base de datos relacional.

Cliente:

- Interfaz desarrollada con Laravel (para alumnos y administradores).
- Permite que los usuarios se registren, consulten actividades y generen inscripciones de forma intuitiva y responsiva.

Servidor:

- Backend construido con Node.js y Express.js, encargado de procesar la lógica de negocio.
- Gestiona las solicitudes del cliente y expone una API REST para la comunicación con la base de datos.
- Implementa validaciones, seguridad (JWT para autenticación) y reglas de negocio (ejemplo: control de cupo en los eventos).

Base de datos:

- PostgreSQL, por su robustez en manejo de datos relacionales y consultas complejas.
- Tablas principales: Usuarios, Eventos, Inscripciones y Notificaciones.
- Garantiza consistencia y evita registros duplicados en inscripciones.

Frameworks y tecnologías

- Lravel : UI moderna, responsiva y multiplataforma.
- Justificación:Una sola base de código para web y móvil, acelerando desarrollo y mantenimiento.

Backend:

- Node.js con Express.js.
- Justificación: Laravel consume datos desde una API, así que el backend puede ser independiente y escalable.

Base de datos:

- PostgreSQL: para eventos, usuarios e inscripciones.

Patrón de diseño

Observer

El patrón Observer es un patrón de diseño de comportamiento que permite que un objeto (llamado sujeto o subject) notifique automáticamente a otros objetos (llamados observadores o observers) cuando su estado cambia. Esto facilita la comunicación entre objetos de manera desacoplada, es decir, sin que los objetos dependan directamente unos de otros.

Qué hace:

- Permite que múltiples componentes reaccionen automáticamente a cambios en otro componente.
- Facilita la actualización en tiempo real de interfaces, reportes o notificaciones cuando ocurren eventos importantes.
- Reduce el acoplamiento entre los módulos, haciendo el sistema más flexible y mantenible.

Para qué sirve en este proyecto:

En la aplicación web de la Universidad Tecnológica de Tecámac, el patrón Observer se utilizará para manejar las notificaciones de inscripciones a actividades. Cada vez que un estudiante se inscribe en un evento, los observadores (correo electrónico, panel administrativo, alertas web) se actualizarán automáticamente sin que el módulo de inscripciones tenga que interactuar directamente con ellos. Esto asegura que los usuarios siempre reciban información actualizada y que el sistema sea más escalable y fácil de mantener.

Cómo se implementa:

Sujeto (Subject): El módulo de inscripciones actúa como sujeto, notificando los cambios cuando un estudiante realiza una inscripción.

Observadores (Observers): Los módulos de notificación (correo electrónico, alertas web y panel administrativo) se suscriben al sujeto y se actualizan automáticamente cuando se registra una inscripción nueva.

Flujo de trabajo:

- El estudiante realiza la inscripción.
- El módulo de inscripciones actualiza su estado.
- Se llama al método notify(), activando a todos los observadores.
- Los observadores envían notificaciones al correo, actualizan el panel administrativo y muestran alertas web.

Beneficios de usar Observer en este proyecto:

- Los estudiantes reciben confirmación inmediata de sus inscripciones.
- La coordinación escolar tiene reportes y estadísticas actualizadas en tiempo real.
- El sistema es más modular, lo que permite agregar nuevas formas de notificación sin modificar el módulo de inscripciones.

Características principales

	ACTIVIDADES	CONTROL	Mes 1					Mes 2				Mes 3				Mes 4	
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	Recolección de datos	PROG.															
		REAL															
2	Análisis	PROG.															
		REAL															
3	Planeación	PROG.															
		REAL															
4	Desarrollo	PROG.															
		REAL															
5	Pruebas	PROG.															
		REAL															

Justificación del versionado del proyecto

Con el propósito de asegurar un desarrollo estructurado, controlado y con un historial verificable, el proyecto utilizará GitHub como plataforma principal de control de versiones.

GitHub proporcionará un entorno eficiente para administrar el código fuente mediante el sistema Git, permitiendo mantener un registro claro y organizado de todas las modificaciones realizadas.

- **Registro de cambios:** Cada actualización del código quedará documentada con información sobre el autor, la fecha y una descripción clara del cambio, lo que facilita el seguimiento del historial y la restauración de versiones previas en caso necesario.
- **Trabajo colaborativo:** Los miembros del equipo podrán trabajar en distintos módulos (frontend, backend y base de datos) de forma simultánea, integrando sus aportaciones sin conflictos y manteniendo la coherencia del proyecto.
- **Gestión de ramas:** Aunque el desarrollo principal se mantendrá en la rama main, se crearán ramas secundarias para el desarrollo y prueba de nuevas características antes de integrarlas al código estable.
- **Transparencia y trazabilidad:** Los commits, issues y pull requests funcionarán como evidencia del progreso del proyecto, de las decisiones técnicas y de la solución de incidencias, garantizando la trazabilidad completa del desarrollo.

Además, se aplicará un sistema de versionado semántico para reflejar la evolución del sistema a lo largo del tiempo. Cada versión representará un avance funcional importante dentro del proyecto:

- **Versión 1.0:** Creación e implementación del repositorio en GitHub con su estructura MVC:
    - **Versión 1.1:** Implementación de página de bienvenida en con la información correspondiente a las Actividades Extracurriculares.
    - **Versión 1.2:** Implementación del Login y Registro de un estudiante para acceder a la plataforma.
    - **Versión 1.2.0:** Mejora de la página de bienvenido, fondo, se agregó responsiva, animaciones y más información de las actividades.

- **Versión 2.0:** Integración de la API de Google, Facebook u otra red social para consumir sus servicios de inicio de sesión.
- **Versión 3.0:** Desarrollo completo del módulo de eventos e inscripciones, con control de cupo y notificaciones automáticas.

Flujo de trabajo utilizado en el proyecto

El flujo de trabajo se basa en la metodología SCRUM, complementada con un modelo de versionamiento por ramas funcionales (branching model) adaptado al esquema de GitHub Flow.
Cada funcionalidad o módulo definido en el backlog del proyecto tiene su propia rama de desarrollo, lo que permite mantener una estructura organizada y controlada durante los sprints.

Ramas funcionales del proyecto:

Rama	**Propósito**	**Responsable**
main	Contiene el código estable y probado. Solo se fusionan cambios aprobados.	Todo el equipo
Feature/	Implementación de la sección de bienvenida inicial.	Frontend
feature/improvedWelcome	Mejoras en la página de bienvenida, incluyendo imágenes, logo más grande y texto justificado.	Fronted
feature/captcha-login	Implementación del sistema CAPTCHA en el login para mayor seguridad.	Frontend / Backend
feature/register	Funcionalidad de registro de nuevos usuarios.	Frontend / Backend
Feature/	Desarrollar una nueva funcionalidad sin romper el código estable, facilitando pruebas, revisiones y colaboración en equipo.	Todo el equipo

Flujo general:

1. Creación del repositorio remoto en GitHub.
2. Clonación del repositorio local en los equipos de desarrollo.
3. Creación de ramas funcionales por módulo o sprint.
4. Desarrollo, pruebas y commits documentados.
5. Validación mediante pull requests y revisión por el líder del proyecto.
6. Fusión en la rama main y generación de una nueva versión liberada.

Este flujo garantiza control, trazabilidad y una integración ordenada de los avances del equipo durante el ciclo de vida del proyecto.

Documentación del versionador (repositorio GitHub)

El repositorio oficial del proyecto se encuentra bajo el nombre “AddedUT” en la plataforma GitHub.

Su estructura y configuración permiten mantener el control del código, la documentación técnica y la sincronización entre los módulos frontend, backend y base de datos.

Configuración principal del repositorio:

1. Creación del repositorio:
    - Nombre: AdeedUT
    - Visibilidad: Pública o privada según requerimientos institucionales.
    - Archivos iniciales: README.md, .gitignore (Node y Laravel), y licencia MIT opcional.
2. Clonación local:
    - git clone https://github.com/Lowdy28/AddedUT.git
    - cd AddedUT
3. Configuración del entorno Git:
    - Registro de nombre y correo del desarrollador:
        - git config --global user.name "Nombre"
        - git config --global user.email "correo@ejemplo.com"
    - Verificación de configuración:
        - git config --list

- Creación de clave SSH para conexión segura:
    - ssh-keygen -t rsa -b 4096 -C "correo@ejemplo.com"
- Subida de clave pública a GitHub:
    - Settings → SSH and GPG Keys

Herramientas necesarias utilizadas en el proceso

Herramienta	Descripción	Fuente / Instalación
Git	Sistema de control de versiones distribuido.	https://github.com
GitHub Desktop (opcional)	Interfaz gráfica para gestionar repositorios GitHub.	https://desktop.github.com
Node.js	Entorno de ejecución para el backend (Express.js).	https://nodejs.org
Laravel SDK	Framework para desarrollo web del frontend.	https://laravel.com/
Mysql	Sistema gestor de base de datos relacional.	https://www.mysql.com/downloads/
VS Code	Editor principal de código.	https://code.visualstudio.com

Tabla de versiones del proyecto

Versión	Fecha de liberación	Módulo o cambio principal	Responsable	Estado
v1.0	30/10/2025	Configuración inicial del repositorio estructurado en MVC.	Programador Backend	Finalizada
v1.1	30/10/2025	Implementación de página de bienvenida en con la información correspondiente a las Actividades Extracurriculares.	Programador Backend	Finalizada

v1.2	31/10/2025	Implementación del Login y Registro de un estudiante para acceder a la plataforma.	Programador Backend	Finalizada
V1.2.0	31/10/2025	Mejora de la página de bienvenido, fondo, se agregó responsiva, animaciones y más información de las actividades.	Programador Backend	Finalizada
v2.1.0	10/10/2025			
v3.0.0	20/10/2025			

Justificación del Certificado de Seguridad utilizado

El proyecto utiliza un certificado SSL/TLS como parte de la configuración del servidor de despliegue, con el objetivo de garantizar la confidencialidad e integridad de la información transmitida entre los usuarios y el servidor web. El uso de HTTPS evita que las credenciales y los datos personales de los estudiantes puedan ser interceptados o modificados durante su transmisión.

Justificación técnica:

- HTTPS (basado en SSL/TLS) cifra la información y autentica el servidor ante el cliente.
- Cumple con los estándares de seguridad recomendados por OWASP (Open Web Application Security Project).
- Permite cumplir con normativas institucionales y políticas de protección de datos.
- Mejora la clasificación del sitio en motores de búsqueda y la confianza de los usuarios.

Evidencia documental:

- Archivo de configuración del hosting o proveedor de nube (ej. Render, Netlify, AWS o Railway) donde se especifica el certificado SSL activo.
- Reporte de auditoría de seguridad (por ejemplo, generado con SSL Labs o Qualys) mostrando la calificación de seguridad del dominio (A o superior).
- Política de seguridad institucional donde se exige el uso de HTTPS en todos los sistemas web.

Justificación del Mecanismo de Autenticación Remota utilizado

El sistema implementa autenticación centralizada mediante tokens JWT y validación con claves secretas almacenadas en variables de entorno. La sesión del usuario no depende de una conexión persistente, sino de un token firmado digitalmente, que permite acceder de manera segura desde cualquier dispositivo conectado a Internet.

Como medida adicional de seguridad, se implementa **CAPTCHA** en el login para prevenir intentos de acceso automatizados por bots, protegiendo así la integridad de las cuentas de usuario.

Justificación técnica:

1. **Autenticación con JWT**
    - Permite que los usuarios se autentiquen desde distintos dispositivos (computadora, tableta, teléfono) sin exponer credenciales.
    - Evita mantener sesiones en memoria del servidor, reduciendo el riesgo de secuestro de sesión.
    - Los tokens se pueden revocar o invalidar si se detecta uso indebido.
    - Compatible con Single Sign-On (SSO) y estándares modernos como OAuth 2.0 / OpenID Connect.
2. **Control de acceso por roles (RBAC)**
    - Diferencia operaciones permitidas para estudiantes y administradores.
    - Protege recursos críticos del sistema como eventos, inscripciones y reportes.
3. **Medida complementaria – CAPTCHA**
    - Evita ataques automatizados (bots) intentando iniciar sesión masivamente.
    - Refuerza la seguridad de la autenticación sin afectar la experiencia de usuarios legítimos.

Evidencia:

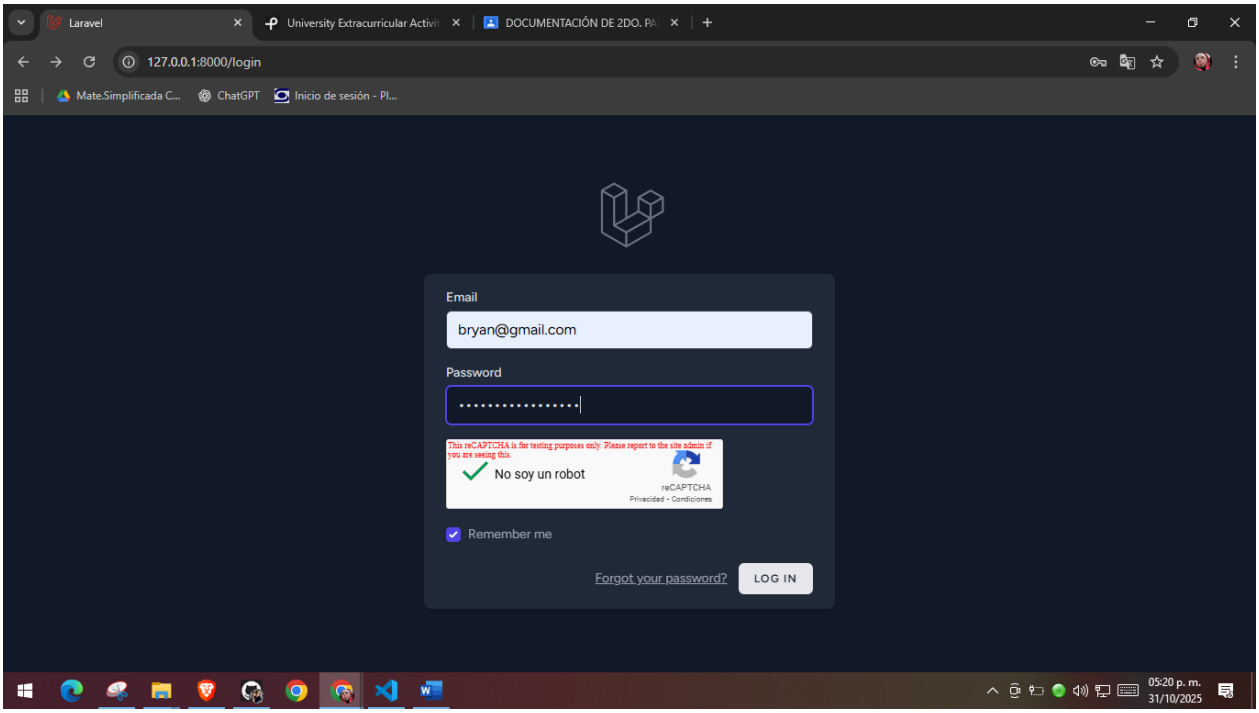

Figura 1.0: Interfaz del Inicio de Sesión con el mecanismo de autentificación CAPTCHA.

Justificación del Mecanismo de Autenticación en Servicios Web (SOAP y/o REST)

El proyecto se basa en servicios web RESTful, en los que cada endpoint se protege mediante autenticación basada en tokens y control de roles. Los usuarios deben estar previamente autenticados para acceder a los recursos del sistema (eventos, inscripciones, reportes). Además, se incluye un control de acceso por rol (RBAC) para diferenciar las operaciones permitidas a estudiantes y administradores.

Justificación técnica:

- “Se eligió la autenticación basada en tokens JWT porque permite sesiones stateless, es decir, no requiere almacenar la sesión en memoria del servidor. Esto es ideal para APIs REST, donde los clientes pueden estar en múltiples dispositivos.
- Además, los tokens son firmados digitalmente, asegurando que no puedan ser alterados. Se pueden revocar o expirar según políticas de seguridad.
- El control de acceso por roles (RBAC) asegura que los estudiantes solo accedan a funcionalidades de inscripción y consulta, mientras que los administradores gestionan eventos y reportes, manteniendo la seguridad y segregación de funciones.”

Evidencia documental:

- Archivo .env con JWT_SECRET y JWT_EXPIRES_IN.
- Middleware en Node.js que valida tokens y roles (authMiddleware.js, por ejemplo).
- Documentación de API (Postman o Swagger) donde se muestran los endpoints protegidos y sus permisos.
- Logs de intentos de acceso fallidos (HTTP 401/403).