necesarias para asegurar la calidad del software antes de su liberación.

Garantizar el cumplimiento de los requerimientos funcionales y no funcionales.

Detectar defectos de diseño o implementación antes del despliegue.

Validar la integración entre frontend, backend y base de datos.

Asegurar la estabilidad, usabilidad y seguridad del sistema.

Alcance

Incluye:

Pruebas funcionales de autenticación, gestión de usuarios, pagos, horas trabajadas y unidades de vivienda.

Validación de la interfaz de usuario (HTML, JS, CSS).

Pruebas de integración entre APIs y frontend.

Verificación de roles y permisos de acceso.

No incluye:

Pruebas de rendimiento a gran escala.

Pruebas de seguridad avanzadas o de penetración.

Integración con pasarelas de pago externas no implementadas.

Tipos de Pruebas Planificadas

Tipo	Objetivo	Técnica	Nivel
Caja Negra (Funcionales)	Validar comportamiento frente a entradas y salidas esperadas.	Pruebas manuales y automatizadas.	Sistema
Caja Blanca (Estructurales)	Verificar la lógica interna, flujos de control y condiciones.	Unit testing con PHPUnit.	Código / Módulo
Caja Negra de Interfaz (UI/UX)	Evaluar usabilidad, validaciones y diseño responsivo. Maze?	Cypress / Selenium.	Interfaz
Integración	Validar comunicación entre servicios (API y BD). Insomnia?	Postman / Scripts automáticos.	API
Regresión	Comprobar que nuevas funcionalidades no afecten anteriores.	Automatizadas.	Global

Estrategia de pruebas

El testing se realizará en fases incrementales, cubriendo primero los módulos críticos y luego los complementarios.

Diseño de casos de prueba basados en los requerimientos.

Ejecución progresiva: pruebas unitarias → integración → sistema.

Automatización parcial de pruebas funcionales repetitivas.

Verificación de interfaz en navegadores principales (Chrome, Firefox, Edge).

Evaluación de calidad según criterios de aceptación definidos.

Entorno de pruebas

Servidor: Linux Rocky 8 (staging).

Base de datos: MySQL 8.0.

Backend: Laravel 10 (PHP 8). 12!

Frontend: HTML5, CSS3, JavaScript (Vue.js).

Herramientas: Postman, PHPUnit, Cypress, GitHub Actions, JMeter.

Navegadores soportados: Chrome, Firefox, Edge.

Recursos y responsabilidades

Rol	Responsable	Tareas
Líder de Pruebas	Brian Barcelo	Planificación, diseño de casos, control de calidad.
Tester QA	Valentina Mendez	Ejecución de pruebas funcionales y UI.
Desarrollador Backend	Ian Miller	Pruebas unitarias e integración de APIs.
DevOps	Equipo Técnico	Mantenimiento del entorno de pruebas.

Casos de pruebas planificados

ID	Módulo	Descripción	Tipo de Prueba	Resultado Esperado
CN-01	Login	Verificar acceso con credenciales válidas	Caja Negra	Acceso concedido al panel principal.
CN-02	Login	Validar rechazo con credenciales inválidas	Caja Negra	Error visible, sin sesión creada.

CN-03	Pagos	Validar creación de comprobante de pago	Caja Negra	Comprobante almacenado en BD.
WB-01	Comprobante Controller	Validar método crearComprobante()	Caja Blanca	Creación correcta, manejo de errores.
UI-01	Interfaz	Validar mensajes de error y responsividad	UI/UX	Validaciones visuales correctas.

Criterios de entrada

Versión del software estable (build entregado).

Ambiente de pruebas configurado y funcional.

Casos de prueba revisados y aprobados.

Datos de prueba disponibles y cargados.

Criterios de salida

100% de casos ejecutados.

Sin defectos críticos abiertos.

Validación de cobertura mínima del 90%.

Aprobación formal del equipo QA y líder técnico.

Cronograma tentativo

Fase	Actividad	Duración	Responsable
1	Diseño de casos de prueba	1 semana	Braian Barcelo
2	Pruebas unitarias	1 semana	Braian Barcelo
3	Pruebas de integración	1 semana	Braian Barcelo
4	Pruebas funcionales	2 semanas	Braian Barcelo
5	Revisión final y cierre	1 semana	Braian Barcelo

Gestión de defectos

Los defectos se registrarán en GitHub Projects bajo el repositorio del sistema.

El flujo de estados será:

Nuevo → Asignado → Corregido → Verificado → Cerrado.

Riesgos y mitigaciones

Riesgo	Impacto	Mitigación

Fallas de comunicación entre módulos	Alto	Pruebas de integración tempranas.
Errores de permisos o roles	Alto	Validaciones específicas por tipo de usuario.
Ambiente no disponible	Medio	Uso de entornos locales alternativos.
Cambios de alcance	Bajo	Actualización del plan y rediseño de casos.

aprobación y seguimiento

El plan será validado por el equipo de desarrollo y aprobado por el líder técnico antes del inicio de las pruebas.

El seguimiento del progreso se realizará mediante reportes semanales y revisión de métricas de cobertura.

4.4.2. Pruebas de Caja Negra

Alcance del Análisis

Ámbito: aplicaciones web y APIs de Zyber.

Componentes analizados: Frontend de usuarios (HTML, CSS, JS), Backoffice (Laravel), Endpoints REST en api.php. se hizo todo en insomnia

Figura 27.

Evidencia extraída de Evidencia extraída de Insomnia – Workspace completo de colecciones de prueba en Insomnia para el proyecto Zyber.

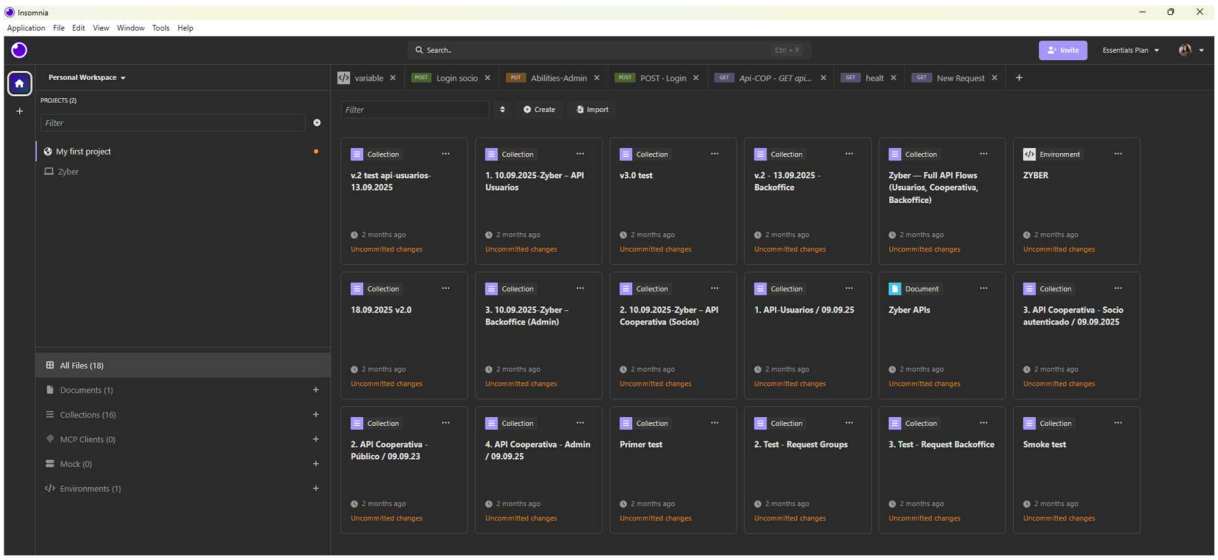
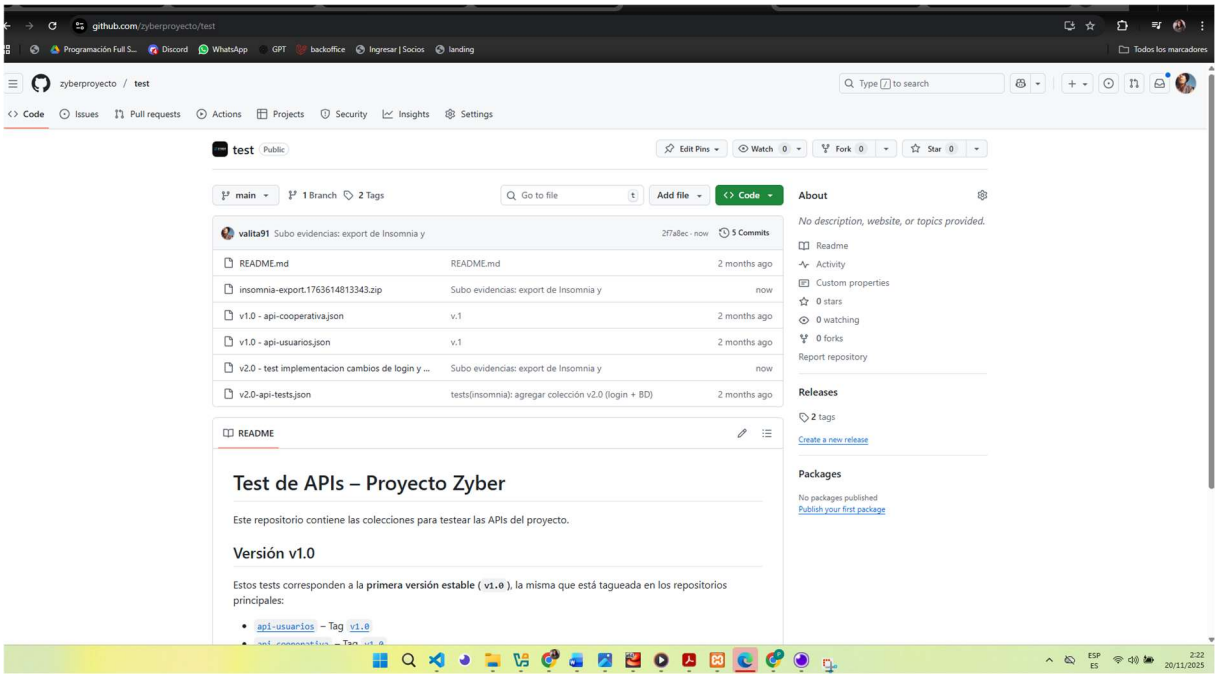


Figura 28.

Evidencia extraída de Evidencia extraída de Insomnia – Repositorio GitHub de pruebas de APIs



Criterios de observación:

Comportamiento funcional, validación de datos, control de acceso, manejo de archivos y respuesta del sistema ante escenarios de uso legítimo y no legítimo.

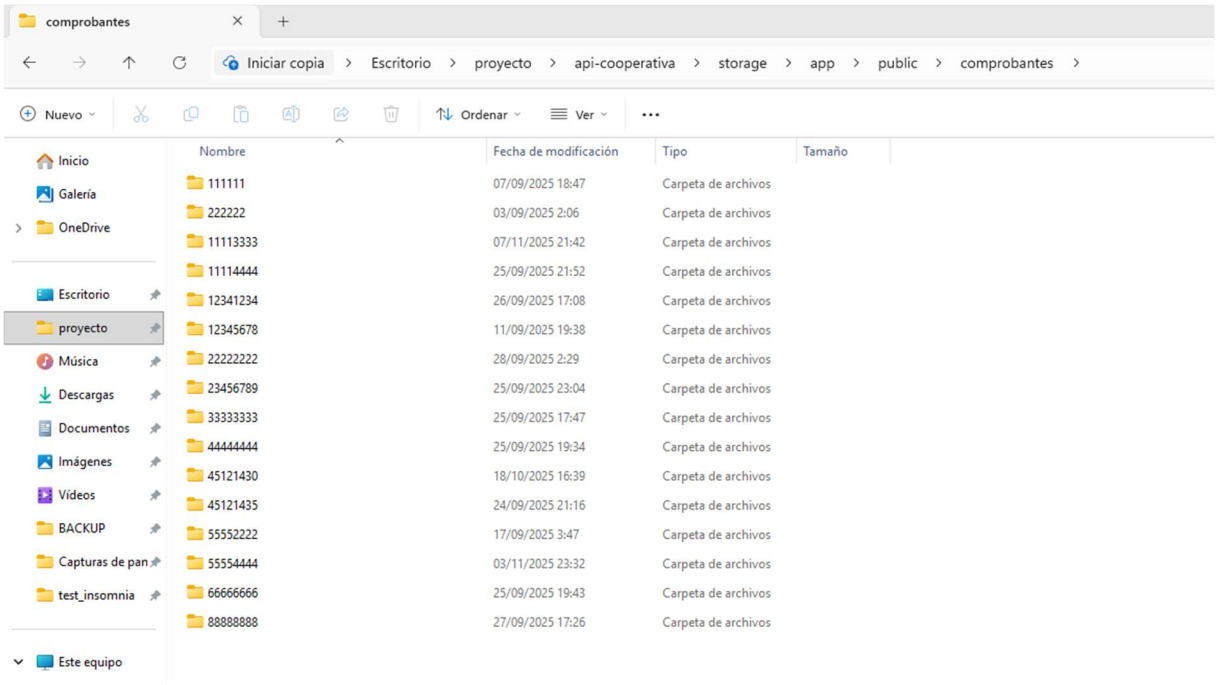
Metodología

El estudio se realizó mediante técnicas de caja negra, sin revisar la lógica interna del código. Se basó en navegación por rutas y vistas, interacción con los endpoints REST y observación de respuestas HTTP, contrastando con los casos de uso definidos para determinar la cobertura funcional.

Arquitectura General Observada

El sistema se estructura bajo una arquitectura cliente-servidor con separación por capas:

Frontend (HTML/JS/CSS) → APIs (Laravel RESTful) → Base de Datos MySQL → Storage de archivos (comprobantes).



Roles principales: Socio (interfaz de usuario) y Administrador (backoffice).

Módulos Identificados

Se detectan los módulos principales: Autenticación, Solicitudes, Comprobantes, Horas, Exoneraciones, Unidades y Backoffice.

Cada uno cuenta con rutas específicas observadas en los controladores y archivos de rutas del proyecto.

Frontend – Comportamiento Observado

El frontend presenta las vistas index.html, perfil.html, pagos.html, estado.html, horas.html y unidad.html.

Cada vista utiliza scripts JS que realizan peticiones a las APIs mediante fetch, autenticadas por token.

Flujos Funcionales

Flujo 1 – Registro y aprobación: usuario se registra, queda pendiente y es aprobado por admin.

Flujo 2 – Gestión de comprobantes: socio sube comprobante, admin aprueba o rechaza.

Flujo 3 – Registro de horas: socio ingresa horas, admin valida.

Flujo 4 – Exoneración: socio solicita, admin resuelve.

Flujo 5 – Asignación de unidad: admin asigna unidad, socio la visualiza.

Casos de Prueba de Caja Negra

Se diseñaron casos de prueba para cada módulo, verificando comportamiento esperado:

CT-01 Login válido: genera token.

CT-02 Login inválido: devuelve 401.

CT-03 Registro duplicado: devuelve 422.

CT-04 Subida válida: 201 Created.

CT-05 Subida inválida: 400 Bad Request.

CT-06 Consulta de estado: datos del usuario.

CT-07 Aprobación de comprobante: cambia estado a aprobado.

CT-08 Registro de horas: éxito.

CT-09 Exceso de horas: error 422.

CT-10 Solicitud de exoneración: crea registro.

CT-11 Duplicación: error controlado.

CT-12 Acceso no autorizado: 403.

CT-13 Aprobación de solicitud: estado actualizado.

CT-14 Visualización unidad: datos correctos.

CT-15 Sin autenticación: 401.

Comportamientos Detectados por Rol

Socio: puede registrar, subir comprobantes, solicitar exoneraciones, ver estado y unidad.

Administrador: aprueba registros, comprobantes, horas, exoneraciones y asigna unidades.

Resultados Generales del Análisis

El sistema Zyber presenta una estructura coherente y segmentada, con control de roles implementado, flujo funcional completo y respuestas HTTP predecibles.

Las rutas observadas se ajustan a los casos de uso definidos y las pruebas evidencian funcionamiento correcto en la mayoría de los módulos.

Cobertura Funcional vs Casos de Uso

CU-01 Registro de socio: Implementado

CU-02 Autenticación: Implementado

CU-03 Perfil: Implementado

CU-04 Aprobación: Implementado

CU-05 Subida de comprobante: Implementado

CU-06 Validación de comprobante: Implementado

CU-07 Registro de horas: Implementado

CU-08 Exoneración: Implementado

CU-09 Asignación de unidad: Implementado