INGENIERÍA DE SOFTWARE III
PARCIAL 3 – PARTE PRÁCTICA

Plan de pruebas - Aplicación UniRide

INTEGRANTES:
A00401993 Juan Andrés Castaño
A00371952 Camilo Pollero Celis
A00405163 Manuela Gonzalez
A00405100 Luis López

1 Introducción

UniRide es una aplicación de movilidad universitaria que busca conectar pasajeros y conductores dentro de la comunidad Icesi. Este plan de pruebas contempla pruebas unitarias, de integración y de sistema sobre los módulos claves, incluyendo registro de usuarios, seguridad, reviews y tarifas.

1.1 Alcance

1.1.1 Dentro del Alcance

Modulo	Responsable	Funcionalidad	Descripción
Registro y Verificación	Juan Andrés Castaño	Registro de usuarios	Usa Factory Method para instanciar Admin, Passenger, Driver
Gestión de Reviews	Luis López	Crear y mostrar reviews	Incluye creación, respuestas y visualización de hilos
Cálculo de Tarifas	Manuela Gonzales	Cálculo según estrategia	Usa Strategy para tarifas por tiempo, distancia, descuentos
Seguridad	Camilo Pollero	Monitoreo y alertas	Ubicación en vivo, botón de emergencia y compartir viaje

1.1.2 Fuera de alcance

No se cubren pruebas de estrés ni pruebas sobre pasarelas de pago reales. Tampoco se incluye validación del login con proveedores externos.

1.2 Objetivo de Calidad

- Asegurar la correcta funcionalidad de los módulos críticos
- Validar que los errores se manejan apropiadamente
- Confirmar que se cumplen los requisitos funcionales
- Garantizar la estabilidad y seguridad del sistema

1.3 Roles y Responsabilidades

Nombre	Módulo	Responsabilidad
Juan Andrés Castaño	Registro	Diseño y ejecución de pruebas Factory Method
Luis López	Reviews	Pruebas de creación y visualización de reseñas
Manuela Gonzales	Tarifas	Pruebas de Strategy y clases equivalentes
Camilo Pollero	Seguridad	Validación de GPS, emergencia y compartir viaje

2 Metodología de prueba

2.1 Niveles de prueba

Se realizan pruebas unitarias por clase, pruebas de integración entre controladores y vistas, y pruebas funcionales sobre los casos de uso principales.

2.2 Clasificación de errores

- Crítico: Bloquea funcionalidades principales
- Medio: Permite continuar pero con degradación
- Menor: No afecta funcionalidad

2.4 Criterios de suspensión y requisitos de reanudación

Se suspenderán las pruebas ante errores críticos que impidan continuar. Se reanudarán al aplicar correcciones.

2.5 Completitud de la prueba

Una prueba se considera completa si se ejecutan todos los casos planificados sin errores críticos sin resolver.

2.6 Tareas del proyecto, estimación y cronograma

Tarea	Miembros	Esfuerzo estimado
Diseño de pruebas individuales	Todos	6h
Ejecución de pruebas unitarias	Todos	5h
Integración y pruebas del sistema	Todos	6h
Análisis y documentación	Todos	4h
Revisión final	Todos	3h

Total: 24 horas-hombre

3 Entregables de prueba

- Plan de pruebas grupal (este documento)
- Diseños individuales (PDF)
- Evidencias de ejecución de pruebas
- Reporte de errores

4 Necesidades de recursos y ambientes

4.1 Herramientas de prueba

Recursos	Descripción
ScalaTest / JUnit	Ejecución de pruebas unitarias
Excel / Google Sheets	Seguimiento de casos y resultados
AllPairs Tool	Generación de combinatorias
GitHub Actions	Integración continua y validación automática

4.2 Entorno de prueba

Sistema operativo: Windows 10 / Ubuntu
Lenguaje: Scala 2.13+
IDE: IntelliJ IDEA
Repositorio: GitHub Classroom con CI