

Estado del Proyecto Dale

Fecha de última actualización: 2025-10-29 15:12:23

Resumen General

Estado:  En Desarrollo - Fase 6 Completada

Progreso: 6/10 fases principales completadas (60%)

Fases Completadas

1. Estructura Base del Monorepo (Completado)

Status:  Completado

Archivos creados:

- `/.spec/constitution.md` (268 líneas) - Reglas SDD y gobernanza
- `/README.md` (335 líneas) - Documentación principal
- `/.gitignore` (65 líneas) - Exclusiones de git
- Directorios: `/frontend`, `/backend`, `/infra`, `/docs`, `/.spec/specs`, `/.spec/plans`, `/.spec/tasks`, `/.spec/evals`

2. Documentación SDD (Completado)

Status:  Completado

Especificaciones creadas:

- `/.spec/specs/ride-mvp.md` (494 líneas) - 10 historias de usuario
- `/.spec/plans/ride-mvp-plan.md` (937 líneas) - Arquitectura técnica
- `/.spec/tasks/ride-mvp-tasks.md` (1006 líneas) - 41 tareas, ~120h
- `/.spec/evals/ride-mvp-evals.md` (538 líneas) - 13 criterios de aceptación

Total documentación SDD: 2,243 líneas

3. Configuración Frontend (Completado)

Status:  Completado

Stack instalado:

- Next.js 14 (App Router) + TypeScript
- TailwindCSS v4
- Supabase Client (`@supabase/supabase-js`)
- next-pwa (PWA support)

Archivos creados:

Configuración:

- `frontend/package.json` - Dependencias del proyecto
- `frontend/tailwind.config.ts` (63 líneas) - Tema personalizado
- `frontend/next.config.ts` - Configuración Next.js + PWA

- `frontend/tsconfig.json` - TypeScript strict mode
- `frontend/public/manifest.json` (24 líneas) - PWA manifest

Componentes UI (Kirk UI wrappers):

- `src/components/ui/Button.tsx` (90 líneas)
- `src/components/ui/Input.tsx` (73 líneas)
- `src/components/ui/Card.tsx` (87 líneas)
- `src/components/ui/Modal.tsx` (112 líneas)
- `src/components/ui/Toast.tsx` (97 líneas)
- `src/components/ui/index.ts` (7 líneas)

Utilidades y contextos:

- `src/lib/utils.ts` (6 líneas) - Función `cn()` para clases
- `src/lib/supabase.ts` (15 líneas) - Cliente Supabase
- `src/lib/api.ts` (84 líneas) - Cliente API tipo-seguro
- `src/contexts/AuthContext.tsx` (97 líneas) - Estado de autenticación

Páginas:

- `src/app/layout.tsx` (41 líneas) - Layout raíz con AuthProvider
- `src/app/page.tsx` (81 líneas) - Landing page
- `src/app/globals.css` (18 líneas) - Estilos globales

Variables de entorno:

- `frontend/.env` - Credenciales Supabase configuradas
- `frontend/.env.example` (9 líneas) - Plantilla de variables

4. Configuración Backend (Completado)

Status:  Completado

Stack configurado:

- FastAPI 0.109
- Prisma ORM (prisma-client-py)
- PostgreSQL (via Supabase)
- Docker + Docker Compose

Archivos creados:

Base del backend:

- `backend/pyproject.toml` (26 líneas) - Dependencias Poetry
- `backend/Dockerfile` (22 líneas) - Contenedor Python
- `backend/.env` - Variables de entorno configuradas
- `backend/.env.example` (12 líneas) - Plantilla

Schema y aplicación:

- `backend/prisma/schema.prisma` (69 líneas) - Modelos User, Ride, Booking
- `backend/app/main.py` (43 líneas) - FastAPI app con CORS

Infraestructura:

- `infra/docker-compose.yml` (24 líneas) - Backend + PostgreSQL local

Herramientas:

- `Makefile` (79 líneas) - Comandos de desarrollo

5. Configuración Supabase y Seeding (Completado) ✨ NUEVO

Status: ☒ Completado

Credenciales configuradas:

- ☒ SUPABASE_URL: <https://sydhgjtsggyglqulxfvh.supabase.co>
- ☒ SUPABASE_ANON_KEY: Configurada en frontend/.env
- ☒ SUPABASE_SERVICE_ROLE_KEY: Configurada en backend/.env
- ☒ DATABASE_URL: Configurada para Prisma

Migración aplicada:

- ☒ Migración `create_dale_tables` ejecutada exitosamente
- ☒ Tablas creadas: User, Ride, Booking
- ☒ Índices de rendimiento creados:
 - `idx_ride_cities_date` (búsqueda por ciudad y fecha)
 - `idx_ride_driver` (búsqueda por conductor)
 - `idx_booking_ride` (búsqueda por viaje)
 - `idx_booking_rider` (búsqueda por pasajero)

Datos de ejemplo insertados:

- ☒ 5 usuarios (3 conductores, 2 pasajeros)
 - María García (conductor) - Madrid/Sevilla
 - Juan Rodríguez (conductor) - Barcelona/Valencia
 - Laura Sánchez (conductor) - Valencia/Madrid
 - Ana Martínez (pasajero)
 - Carlos López (pasajero)
- ☒ 5 viajes programados
 - Madrid → Barcelona (mañana, 3 plazas, €25)
 - Barcelona → Valencia (2 días, 2/4 plazas, €18.50)
 - Madrid → Sevilla (3 días, 2 plazas, €30)
 - Valencia → Madrid (5 días, 3 plazas, €22)
 - Barcelona → Madrid (mañana, 1/4 plazas, €28)
- ☒ 4 reservas activas
 - Ana: Barcelona → Valencia (confirmada)
 - Carlos: Barcelona → Valencia (confirmada)
 - Ana: Barcelona → Madrid (pendiente)
 - Carlos: Barcelona → Madrid (confirmada)

Archivos creados:

- `backend/scripts/seed.py` (232 líneas) - Script de seeding Python
- `backend/scripts/__init__.py` - Módulo Python

Verificación de conectividad: ☒ Exitosa






6. Implementación de API Backend (Completado) ✨ NUEVO

Status:  Completado

Prioridad: Alta

Dependencias: Fase 5 

Tareas completadas:

-  **TASK-16:** Middleware de autenticación JWT implementado
 - Validación de tokens de Supabase
 - Funciones `get_current_user`, `get_current_user_optional`, `require_role`
 - Manejo de errores 401 (token expirado/inválido) y 403 (permisos)
-  **TASK-17:** Endpoints de Rides implementados
 - POST `/api/rides` - Crear viaje (solo conductores)
 - GET `/api/rides` - Buscar con filtros (`from_city`, `to_city`, `date`, `min_seats`, `max_price`)
 - GET `/api/rides/{id}` - Detalle de viaje
 - GET `/api/rides/my/rides` - Mis viajes como conductor
 - DELETE `/api/rides/{id}` - Eliminar mi viaje
-  **TASK-18:** Endpoints de Bookings implementados
 - POST `/api/bookings` - Reservar plaza (decrementa `seats_available`)
 - GET `/api/bookings` - Mis reservas
 - GET `/api/bookings/{id}` - Detalle de reserva
 - DELETE `/api/bookings/{id}` - Cancelar reserva (incrementa `seats_available`)
 - PATCH `/api/bookings/{id}/confirm` - Confirmar reserva (solo conductor)
-  **TASK-19:** Endpoints de perfil implementados
 - GET `/api/me` - Obtener perfil autenticado
 - PATCH `/api/me` - Actualizar perfil
 - GET `/api/users/{id}` - Perfil público de usuario
-  **TASK-20:** Documentación OpenAPI
 - Schema automático de FastAPI
 - Swagger UI en `/docs`
 - ReDoc en `/redoc`
 - Documentación completa en `backend/API_DOCS.md` (285 líneas)

Archivos creados:

Modelos y schemas:

- `backend/app/models/schemas.py` (118 líneas) - Modelos Pydantic
 - `UserBase`, `UserCreate`, `UserUpdate`, `UserResponse`
 - `RideBase`, `RideCreate`, `RideResponse`, `RideSearchParams`
 - `BookingBase`, `BookingCreate`, `BookingResponse`
 - `TokenPayload`, `ErrorResponse`

Middleware y utilidades:

- `backend/app/middleware/auth.py` (114 líneas) - Autenticación JWT
- `backend/app/utils/database.py` (44 líneas) - Cliente Supabase singleton

Rutas de API:

- `backend/app/routes/users.py` (99 líneas) - 3 endpoints
- `backend/app/routes/rides.py` (221 líneas) - 5 endpoints

- `backend/app/routes/bookings.py` (282 líneas) - 5 endpoints

Aplicación principal:

- `backend/app/main.py` (161 líneas) - FastAPI app actualizada
 - CORS configurado
 - Routers registrados
 - Manejo de errores personalizado
 - Lifespan events

Documentación:

- `backend/API_DOCS.md` (285 líneas) - Documentación completa de endpoints

Testing:

- `backend/test_api.py` (115 líneas) - Script de pruebas básicas

Validaciones implementadas:

- ☒ Email con regex pattern
- ☒ Coordenadas geográficas (lat: -90 a 90, lon: -180 a 180)
- ☒ Fechas futuras para viajes
- ☒ Asientos entre 1 y 8
- ☒ Precios no negativos
- ☒ Longitud de strings (nombres, notas)

Lógica de negocio:

- ☒ Reserva decrementa `seats_available`
- ☒ Cancelación incrementa `seats_available`
- ☒ No puedes reservar tu propio viaje
- ☒ No puedes tener reservas duplicadas
- ☒ Solo conductor puede confirmar reservas
- ☒ Solo creador puede eliminar viajes
- ☒ Verificación de permisos para acceder a reservas

Servidor ejecutándose: ☒ Verificado

- Puerto 8000
- CORS configurado para localhost:3000 y Vercel
- Health check respondiendo



Fases Pendientes

7. Implementación de Páginas Frontend

- `backend/app/models/` - Modelos Pydantic para validación

Estimación: 8-10 horas

7. Implementación de Páginas Frontend

Status: ● Pendiente

Prioridad: Alta

Dependencias: Fase 6

Tareas:

- ☐ **TASK-21:** Página de búsqueda de viajes
 - `/rides` - Búsqueda con filtros (desde, hasta, fecha)
 - Lista de resultados con RideCard
 - Filtros reactivos
- ☐ **TASK-22:** Página de publicar viaje
 - `/offer` - Formulario de creación (solo drivers)
 - Validación de campos
 - Integración con Google Maps para coordenadas (opcional)
- ☐ **TASK-23:** Página de mis reservas
 - `/bookings` - Lista de reservas del usuario
 - Cancelar reserva
 - Estados: pending, confirmed, cancelled
- ☐ **TASK-24:** Página de perfil
 - `/profile` - Editar nombre, avatar
 - Cambiar rol (rider/driver)
- ☐ **TASK-25:** Páginas de autenticación
 - `/login` - Formulario de login
 - `/signup` - Formulario de registro
 - Redirección tras autenticación

Componentes a crear:

- `components/RideCard.tsx` - Tarjeta de viaje
- `components/RideForm.tsx` - Formulario crear/editar viaje
- `components/BookingCard.tsx` - Tarjeta de reserva
- `components/FiltersBar.tsx` - Barra de filtros
- `components/Header.tsx` - Navegación
- `components/BottomNav.tsx` - Navegación móvil

Estimación: 12-15 horas

8. Testing

Status: ● Pendiente

Prioridad: Media

Dependencias: Fase 6, 7

Tareas:

- ☐ **TASK-26:** Tests E2E con Playwright
 - Flujo de autenticación (signup, login, logout)
 - Búsqueda de viajes
 - Reservar y cancelar plaza
 - Crear viaje (driver)
- ☐ **TASK-27:** Tests backend con pytest
 - Tests de API endpoints
 - Tests de lógica de negocio
 - Tests de validación de datos

- Cobertura >80%

Archivos a crear:

- `frontend/tests/e2e/*.spec.ts` - Tests Playwright
- `backend/tests/test_*.py` - Tests pytest
- `frontend/playwright.config.ts` - Configuración Playwright
- `backend/pytest.ini` - Configuración pytest

Estimación: 8-10 horas

9. CI/CD

Status: ● Pendiente

Prioridad: Media

Dependencias: Fase 8

Tareas:

- ☐ **TASK-28:** Workflow de spec-gate
 - Verificar que specs están actualizadas
 - Bloquear merge si specs desactualizadas
- ☐ **TASK-29:** Workflow de testing
 - Lint (ESLint, Ruff)
 - Typecheck (TypeScript, mypy)
 - Tests (Playwright, pytest)
- ☐ **TASK-30:** Workflow de deployment
 - Deploy frontend a Vercel
 - Deploy backend a Railway/Supabase Edge Functions

Archivos a crear:

- `.github/workflows/spec-gate.yml`
- `.github/workflows/test.yml`
- `.github/workflows/deploy.yml`

Estimación: 4-6 horas

10. PWA y Optimización

Status: ● Pendiente

Prioridad: Baja

Dependencias: Fase 7

Tareas:

- ☐ **TASK-31:** Completar PWA
 - Generar iconos (192x192, 512x512)
 - Implementar service worker
 - Probar instalación en móvil
- ☐ **TASK-32:** Auditoría de accesibilidad
 - Ejecutar axe accessibility
 - Corregir problemas de contraste
 - ARIA labels

- ☐ **TASK-33:** Optimización de rendimiento
 - Optimizar imágenes con Next Image
 - Code splitting
 - Bundle analysis
 - Alcanzar Lighthouse >90

Estimación: 6-8 horas

11. Documentación con MkDocs

Status: ● Pendiente

Prioridad: Baja

Dependencias: Todas las anteriores

Tareas:

- ☐ **TASK-34:** Configurar MkDocs
 - Instalar mkdocs-material
 - Crear estructura de docs
- ☐ **TASK-35:** Escribir documentación
 - Welcome - Introducción al proyecto
 - SDD Philosophy - Metodología
 - Setup - Instalación y configuración
 - Frontend Guide - Desarrollo frontend
 - Backend Guide - Desarrollo backend
 - Auth - Sistema de autenticación
 - Rides & Bookings - Lógica de negocio
 - PWA - Progressive Web App
 - Deployment - Despliegue

Estimación: 6-8 horas



Próximos Pasos Inmediatos


- ☒ ~~Configurar Supabase y aplicar migraciones~~ (COMPLETADO)
- ☒ ~~Poblar base de datos con datos de ejemplo~~ (COMPLETADO)
- ☒ ~~Implementar endpoints de API en FastAPI~~ (COMPLETADO)
- SIGUIENTE:** Implementar páginas frontend de Dale (Fase 7)
 - Página de búsqueda de viajes (/rides)
 - Página de publicar viaje (/offer)
 - Página de mis reservas (/bookings)
 - Página de perfil (/profile)
 - Páginas de autenticación (/login, /signup)




Hitos del Proyecto

- ☒ ~~Milestone 1: Fundación SDD y estructura (Fases 1-2)~~ ☒ Completado
- ☒ ~~Milestone 2: Frontend y Backend scaffolding (Fases 3-4)~~ ☒ Completado
- ☒ ~~Milestone 3: Base de datos productiva (Fase 5)~~ ☒ Completado

✓ **Milestone 4:** API Backend completa (Fase 6) -  Completado

☐ **Milestone 5:** Frontend completo (Fase 7) -  En espera

☐ **Milestone 6:** Producción (Fases 8-11) -  En espera

Métricas del Proyecto

Líneas de código escritas: ~7,000+

Líneas de especificación SDD: 2,243

Componentes UI: 5 (Button, Input, Card, Modal, Toast)

Páginas frontend: 1 (landing) + 5 pendientes

Endpoints API: 13 implementados (3 users, 5 rides, 5 bookings)

Modelos Pydantic: 13 schemas de validación

Middleware: JWT authentication + CORS

Cobertura de tests: 0% (pendiente Fase 8)

Tablas de base de datos: 3 (User, Ride, Booking)

Datos de ejemplo: 5 usuarios, 5 viajes, 4 reservas

Notas Técnicas

Node.js version warning: El entorno tiene Node.js v18.19.0 pero Next.js 16 requiere $\geq 20.9.0$. Funciona pero con advertencia. Considerar upgrade para producción.

FastAPI con Supabase: Se usa el cliente Python de Supabase directamente en lugar de Prisma para aprovechar mejor las features de Supabase (Auth, RLS, etc.).

PWA manifest: Creado pero faltan iconos reales. Usar herramientas como <https://realfavicongenerator.net/>

Kirk UI: Componentes personalizados replicando Kirk UI. Si BlaBlaCar abre el paquete oficial, migrar.

JWT Secret: Actualmente usando placeholder. En producción, obtener el JWT secret real de Supabase dashboard.

Enlaces Útiles

- **Supabase Dashboard:** <https://supabase.com/dashboard/project/sydhgjtsggyglqlxfvh>
 - **API Swagger UI:** <http://localhost:8000/docs>
 - **API ReDoc:** <http://localhost:8000/redoc>
 - **API Docs (Markdown):** `/backend/API_DOCS.md`
 - **Spec Constitution:** `/.spec/constitution.md`
 - **Task List:** `/.spec/tasks/ride-mvp-tasks.md`
-

Última actualización: 2025-10-29 15:12:23

Por: MiniMax Agent