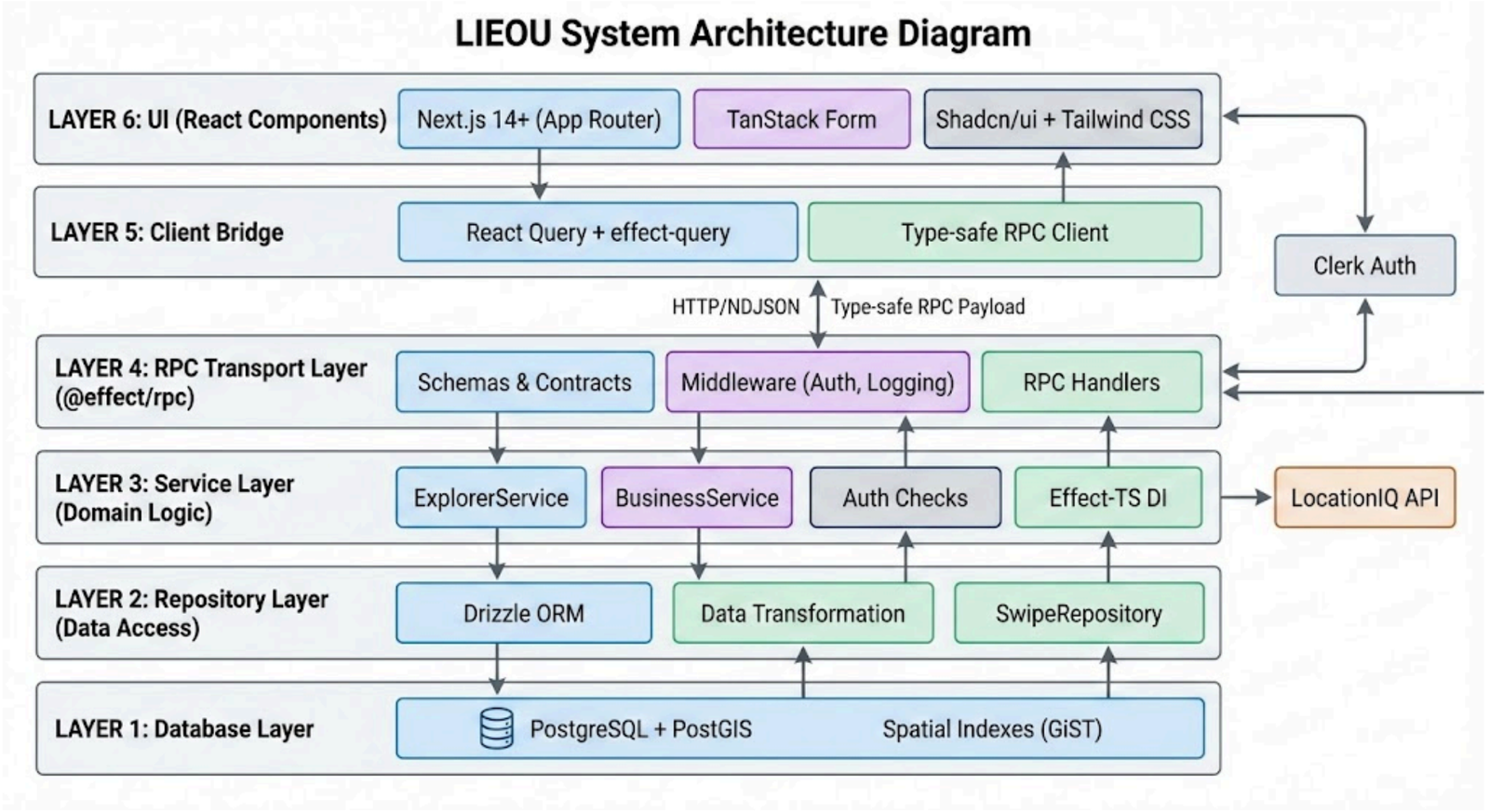
MISOSTASIS

EQUIPO

NOMBRES:

- Julian Alejandro Archila Caro
- Nicolas Alexander
- Santiago Montealegre Romero


LIEOU System Architecture Diagram
LAYER 6: UI (React Components)
Next.js 14+ (App Router)
TanStack Form
Shadcn/ui + Tailwind CSS
LAYER 5: Client Bridge
React Query + effect-query
Type-safe RPC Client
Clerk Auth
HTTP/NDJSON
Type-safe RPC Payload
LAYER 4: RPC Transport Layer (@effect/rpc)
Schemas & Contracts
Middleware (Auth, Logging)
RPC Handlers
LAYER 3: Service Layer (Domain Logic)
ExplorerService
BusinessService
Auth Checks
Effect-TS DI
LocationIQ API
LAYER 2: Repository Layer (Data Access)
Drizzle ORM
Data Transformation
SwipeRepository
LAYER 1: Database Layer
PostgreSQL + PostGIS
Spatial Indexes (GiST)

ESTRUCGURA DE CARPETAS

```
src/
├── app/                          # Next.js routes & UI
│   ├── rpc/route.ts              # RPC endpoint (thin wrapper)
│   ├── explorer/                 # Explorer features
│   └── business/                 # Business features
├── server/
│   ├── db/
│   │   ├── schema.ts             # Drizzle schema definitions
│   │   └── service.ts            # DB service with DI
│   ├── repositories/             # Data access (DB queries)
│   ├── services/                 # Business logic
│   ├── rpc/
│   │   ├── groups/               # RPC contract definitions
│   │   ├── handlers/             # RPC implementations
│   │   ├── middlewares/          # Cross-cutting concerns
│   │   └── server.ts             # RPC server setup
│   ├── schemas/                  # Shared types & validation
│   └── utils/                    # Shared utilities (auth, etc.)
├── lib/
│   └── effect-query.ts           # RPC client + React Query bridge
├── data-access/                  # Query/mutation options
└── components/                   # Reusable UI components
```

DESAFIOS TECNICOS

APRENDIZAJES

- *Dependency injection*
- *Effect*
- *Testing para react*

RECOMENDACIONES

- CI/CD - github actions
- Deployment
- Diseño de APIS (Desde el inicio)

Gracias por su atención!