Web-Tier Boundary Contract

Overview

The UPRISE_NEXT platform enforces a strict architectural boundary between the **Web Tier** and the **Data Tier**. This document describes the contract, enforcement mechanisms, and best practices.

The Contract

✅ Web Tier CAN:

- Make HTTP requests to API endpoints
- Connect to Socket.IO server for real-time updates
- Use client-side state management (Zustand, React Query)
- Perform client-side validation
- Render UI components
- Handle user interactions
- Cache API responses

❌ Web Tier CANNOT:

- Import `@prisma/client` or any database libraries
- Directly connect to databases
- Access environment variables containing database credentials
- Instantiate database clients
- Execute raw SQL queries
- Access data models that contain database connection logic

Why This Boundary Exists

1. **Security**: Prevents exposure of database credentials in client-side code
2. **Scalability**: Allows independent scaling of web and API tiers
3. **Deployment Flexibility**: Web tier (Vercel) separate from API tier (Fly.io)
4. **Clear Separation of Concerns**: Enforces clean architecture
5. **Performance**: Enables edge deployment without database connections

Enforcement Mechanisms

1. Runtime Guards

The `web-tier-guard.ts` module provides runtime protection:

```
import { assertNotWebTier, guardPrismaClient } from '@/lib/web-tier-guard';

// Throws error if called from web tier
function dangerousDbOperation() {
  assertNotWebTier('dangerousDbOperation');
  // ... db code
}
```

2. TypeScript Types

Type definitions prevent database imports at compile time:

```
// This will cause TypeScript error in web tier
import { PrismaClient } from '@prisma/client'; // ❌ Error: never type
```

3. ESLint Rules

ESLint configuration blocks database imports:

```
{
  "rules": {
    "no-restricted-imports": [
      "error",
      {
        "paths": [
          {
            "name": "@prisma/client",
            "message": "🚨 WEB-TIER BOUNDARY VIOLATION"
          }
        ]
      }
    ]
  }
}
```

4. Next.js Middleware

Middleware checks every request for boundary violations:

```
// Runs on every request
export function middleware(request: NextRequest) {
  // Checks for suspicious headers/patterns
  // Adds security headers
}
```

Best Practices

✅ DO: Use API Client

```
// ✅ CORRECT: Use API client in web tier
import { api } from '@/lib/api';

const communities = await api.get('/communities/nearby', {
  params: { lat, lng, radius }
});
```

❌ DON’T: Import Prisma

```
// ❌ WRONG: Never import Prisma in web tier
import { PrismaClient } from '@prisma/client'; // Will throw error

const prisma = new PrismaClient(); // Forbidden!
```

✅ DO: Use Socket.IO for Real-Time

```
// ✅ CORRECT: Use Socket.IO for real-time updates
import { socket } from '@/lib/socket';

socket.emit('join-community', { communityId });
```

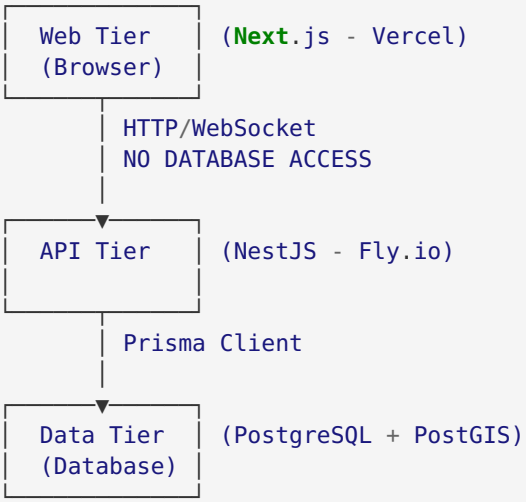
✅ DO: Validate on Both Sides

```
// ✅ CORRECT: Client-side validation for UX
const schema = z.object({
  name: z.string().min(3),
  lat: z.number(),
  lng: z.number(),
});

const validated = schema.parse(formData);

// Then send to API which validates again
await api.post('/communities', validated);
```

Data Flow



Testing the Boundary

See the test suite in `apps/web/__tests__/web-tier-guard.test.ts`:

```
# Run web-tier boundary tests
pnpm --filter web test
```

Error Messages

When a boundary violation occurs, you'll see:

```
╔═══════════════════════════════════════════════════════════╗
║                WEB-TIER BOUNDARY VIOLATION                ║
╠═══════════════════════════════════════════════════════════╣
║ Attempted to import @prisma/client in web tier            ║
║                                                           ║
║ The web application tier CANNOT access the database directly.  ║
║ All data operations must go through the API layer.        ║
╚═══════════════════════════════════════════════════════════╝
```

Troubleshooting

"Cannot import @prisma/client"

✅ **Solution**: Use the API client instead:

```
import { api } from '@/lib/api';
```

"Attempted to instantiate Prisma Client"

✅ **Solution**: Move database logic to API tier (apps/api)

“Database access forbidden”

✅ **Solution**: Create an API endpoint and call it from web tier

Related Documentation

- API Documentation (../api/README.md)
- Socket.IO Documentation (../socket/README.md)
- Architecture Overview (../../README.md)