Propósito

Wrapper robusto para requests HTTP con autenticación, rate limiting y caching.

🗂️ Estructura de Módulos

🔐 Gestión de Sesión

```javascript
export function setSession({ accessToken, user, role }) {
  storage.set(TOKEN_KEY, accessToken, true); // sessionStorage
  storage.set(USER_KEY, user, true);
  storage.set(ROLE_KEY, role, true);
  setState({ user, role });
}
```

⚡ Rate Limiting

```javascript
// Máximo 2 requests por segundo por endpoint
const requestQueue = new Map();


if (lastRequest && now - lastRequest.lastRequest < 500) {
  lastRequest.count++;
  if (lastRequest.count > 5) {
    console.warn(`[RATE-LIMIT] demasiadas requests a ${path}`);
```

```
    }
}
```

💾 Sistema de Caching

```
javascript
```

```
    const responseCache = new Map();


    // Cache automático con TTL configurable
    if (method === "GET" && cache) {
      responseCache.set(cacheKey, {
        data,
        timestamp: Date.now(),
        ttl: cacheTTL
      });
}
```

📡 APIs Específicas

AuthAPI

```
javascript
```

```
    export const AuthAPI = {
      googleLogin(credential) {
        return request("/auth/google", { method: "POST", body: {
credential } });
      }
```

```
};
```

ConfigAPI

```
javascript
```

```
export const ConfigAPI = {
  getGeneralConfig(),        // Todos los autenticados
  getAdvancedConfig(),       // Solo admin
  updateAdvancedConfig(),
  getMQTTStatus(),
  // ... más endpoints de administración
};
```

DevicesAPI

```
javascript
```

```
export const DevicesAPI = {
  getAllDevices(),
  getDeviceById(deviceId),
  getDeviceSensorData(deviceId, limit),
  getHistoricalData(deviceId, limit, timeRange)
};
```

GatewayAPI

```
javascript
```

```
export const GatewayAPI = {
  getAllGateways(),
```

```
    getGatewayById(gatewayId),
    getAllEndpoints(),
    getSystemStatus(),          // Con cache de 5 segundos
    getThresholds(),
    updateThresholds(thresholds)
};
```

🛡️ Características de Seguridad

Manejo de Tokens

```
javascript
  const headers = { "Content-Type": "application/json" };
  if (auth) {
    const t = getToken();
    if (t) headers["Authorization"] = `Bearer ${t}`;
}
```

Validación de Sesión

```
javascript
  export async function initSession() {
    // Verificar token válido haciendo request de prueba
    await request("/config/general", { auth: true });
    // Restaurar sesión si es válido
}
```

Limpieza Segura

```javascript
export function clearSession() {
  // Limpiar sessionStorage y localStorage
  storage.del(TOKEN_KEY, true);
  storage.del(TOKEN_KEY, false);
  setState({ user: null, role: ROLES_CONST.GUEST });
}
```