ionpy – PWA Mobile App Roadmap

Monitoring Companion für Handy & Tablet

Erstellt: 2026-02-27
Status: In Planung
Abhängigkeiten: Kein Blocker aus dem Wizard-Backlog – kann sofort gestartet werden.

Architektur-Entscheidung

Die Mobile App ist eine **Progressive Web App (PWA)** die über `/mobile` erreichbar ist. Sie teilt sich den WebSocket-Kanal und den `globalStore` mit der Desktop-App – es gibt **keinen neuen Backend-Code** für das Monitoring-Grundgerüst.

Später kann die PWA mit **Capacitor** in eine echte App (iOS/Android) gewrappet werden ohne den Vue-Code neu schreiben zu müssen.

```
Benutzer öffnet: http://[server-ip]:8000/mobile
                              ↓
                    FastAPI liefert mobile.html
                              ↓
                    Vue 3 App startet (ES Modules)
                              ↓
                    store.js → WebSocket /ws
                              ↓
                    Gleiche Daten wie Desktop-App
```

Technologie-Stack

Komponente	Technologie	Begründung
Framework	Vue 3 (Composition)	Bereits vorhanden, kein Overhead
Charts	uPlot	Bereits vorhanden, sehr performant
State	store.js	1:1 wiederverwendet
Icons	MDI	Bereits vorhanden
CSS	theme.css Variablen	Bereits vorhanden
PWA	Web App Manifest	Kein Framework nötig
Offline	Service Worker	Nur Static Assets cachen
Deployment	FastAPI Static	Eine neue Route

Blocker-Analyse

Einziger echter Blocker: Server-Route

Datei: `web/server.py`
Aufwand: 5 Minuten
Priorität: Muss zuerst

```
# web/server.py — nach der bestehenden read_index() Route ergänzen:

@app.get("/mobile")
async def read_mobile():
    """Einstiegspunkt für die PWA Mobile App."""
    mobile_path = os.path.join(static_dir, "mobile.html")
    if os.path.exists(mobile_path):
        return FileResponse(mobile_path)
    return JSONResponse(
        status_code=404,
        content={"error": "mobile.html nicht gefunden. "
                          "Bitte in static/ ablegen."}
    )

# Service Worker muss vom Root-Scope aus erreichbar sein
# damit er alle /mobile/* Requests cachen kann
@app.get("/sw.js")
async def serve_service_worker():
    sw_path = os.path.join(static_dir, "mobile", "sw.js")
    if os.path.exists(sw_path):
        return FileResponse(
            sw_path,
            media_type="application/javascript",
            headers={"Service-Worker-Allowed": "/"}
        )
    return JSONResponse(status_code=404, content={"error": "sw.js nicht
gefunden"})
```

Alle anderen Backend-Endpoints (`GET /api/devices`, `POST /api/control`, `POST /api/emergency`, `WebSocket /ws`) funktionieren bereits ohne Änderungen.

PHASE 0: Vorbereitung

(~0.5 Tage)

0.1 Dateistruktur anlegen

```
static/
├── index.html                      ← Desktop – unverändert
├── mobile.html                     ← PWA Entry Point
└── mobile/
    ├── manifest.json               ← PWA Manifest (Homescreen Install)
    ├── sw.js                       ← Service Worker (Offline Cache)
    ├── app.js                      ← Vue Root App
    ├── views/
    │   ├── overview.js             ← Geräteübersicht (Startseite)
    │   ├── device_detail.js        ← Einzelgerät Detail
    │   └── alarms.js               ← Alarm-Feed
    └── components/
        ├── device_card.js          ← Gerätekarte für Übersicht
        ├── sensor_tile.js          ← Einzelner Messwert
        ├── mini_chart.js           ← uPlot Sparkline (60s Verlauf)
        └── connection_banner.js    ← WS-Status Anzeige
```

Icons werden benötigt (PNG, beliebig erstellen oder aus Emoji):

```
static/images/
├── icon-192.png    ← App Icon 192x192px (für Android)
├── icon-512.png    ← App Icon 512x512px (für Splash Screen)
└── badge-72.png    ← Notification Badge 72x72px (optional)
```

0.2 PWA Manifest

Datei: `static/mobile/manifest.json` (NEU)

```
{
    "name": "ionpy Pro",
    "short_name": "ionpy",
    "description": "Hardware Monitor & Control",
    "start_url": "/mobile",
    "scope": "/mobile",
    "display": "standalone",
    "background_color": "#18181b",
    "theme_color": "#18181b",
    "orientation": "any",
    "icons": [
        {
            "src": "/static/images/icon-192.png",
            "sizes": "192x192",
            "type": "image/png"
        },
        {
            "src": "/static/images/icon-512.png",
```

```
            "sizes": "512x512",
            "type": "image/png",
            "purpose": "maskable"
        }
    ],
    "categories": ["utilities", "productivity"]
}
```

0.3 Service Worker (Basis)

Datei: `static/mobile/sw.js` (NEU)

```
/**
 * ionpy PWA Service Worker
 * Strategie: Cache-First für statische Assets,
 *            Network-Only für API und WebSocket
 */

const CACHE_NAME = 'ionpy-mobile-v1';

// Diese Dateien werden beim Install gecacht
// → App lädt auch ohne Netzwerk (nur UI, keine Live-Daten)
const STATIC_ASSETS = [
    '/mobile',
    '/static/mobile/app.js',
    '/static/mobile/views/overview.js',
    '/static/mobile/views/device_detail.js',
    '/static/mobile/views/alarms.js',
    '/static/mobile/components/device_card.js',
    '/static/mobile/components/sensor_tile.js',
    '/static/mobile/components/mini_chart.js',
    '/static/mobile/components/connection_banner.js',
    '/static/vendor/vue.global.prod.js',
    '/static/vendor/uPlot.iife.min.js',
    '/static/vendor/uPlot.min.css',
    '/static/vendor/mdi/css/materialdesignicons.min.css',
    '/static/vendor/fonts/jetbrains-mono.css',
    '/static/css/theme.css',
    '/static/js/store.js',
    '/static/images/icon-192.png',
];

// INSTALL: Assets vorab cachen
self.addEventListener('install', event => {
    event.waitUntil(
        caches.open(CACHE_NAME)
            .then(cache => cache.addAll(STATIC_ASSETS))
            .then(() => self.skipWaiting())
    );
```

```
});

// ACTIVATE: Alte Caches aufräumen
self.addEventListener('activate', event => {
    event.waitUntil(
        caches.keys().then(keys =>
            Promise.all(
                keys
                    .filter(k => k !== CACHE_NAME)
                    .map(k => caches.delete(k))
            )
        ).then(() => self.clients.claim())
    );
});

// FETCH: Cache-Strategie
self.addEventListener('fetch', event => {
    const url = new URL(event.request.url);

    // API-Calls IMMER live – nie aus Cache!
    if (url.pathname.startsWith('/api/')) return;

    // WebSocket kann der SW sowieso nicht abfangen
    if (url.protocol === 'ws:' || url.protocol === 'wss:') return;

    // Statische Assets: Cache-First
    event.respondWith(
        caches.match(event.request).then(cached => {
            if (cached) return cached;
            // Nicht im Cache → Netzwerk und danach cachen
            return fetch(event.request).then(response => {
                if (response.ok) {
                    const clone = response.clone();
                    caches.open(CACHE_NAME).then(c => c.put(event.request,
clone));
                }
                return response;
            });
        })
    );
});
```

PHASE 1: Entry Point & Navigation

(~1 Tag)

1.1 mobile.html – Entry Point

Datei: `static/mobile.html` (NEU)

Wichtige Implementierungshinweise für KI:

- `viewport` mit `user-scalable=no` – verhindert ungewolltes Zoomen
- `height: 100dvh` statt `100vh` – dvh respektiert den

mobilen Browser-Chrome (Adressleiste die ein/ausblendet)

- `env(safe-area-inset-*)` für Notch und Home-Indicator (iPhone)
- `touch-action: manipulation` eliminiert den 300ms Tap-Delay
- Alle Vendor-Dateien aus `/static/vendor/` – identisch zur Desktop-App

```
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
                                   maximum-scale=1.0, user-scalable=no,
                                   viewport-fit=cover">
    <meta name="apple-mobile-web-app-capable" content="yes">
    <meta name="apple-mobile-web-app-status-bar-style" content="black-
translucent">
    <meta name="theme-color" content="#18181b">
    <link rel="manifest" href="/static/mobile/manifest.json">
    <title>ionpy Mobile</title>

    <!-- Gleiche Vendor-Libs wie Desktop -->
    <link rel="stylesheet" href="/static/vendor/fonts/jetbrains-mono.css">
    <link rel="stylesheet"
href="/static/vendor/mdi/css/materialdesignicons.min.css">
    <link rel="stylesheet" href="/static/vendor/uPlot.min.css">
    <link rel="stylesheet" href="/static/css/theme.css">
    <script src="/static/vendor/vue.global.prod.js"></script>
    <script src="/static/vendor/uPlot.iife.min.js"></script>

    <style>
        /* ===== MOBILE RESET ===== */
        *, *::before, *::after {
            box-sizing: border-box;
            -webkit-tap-highlight-color: transparent;
            touch-action: manipulation;
        }

        body {
            background: var(--bg-level-0);
            color: var(--text-main);
            font-family: var(--font-main);
            margin: 0;
```

```
            overflow: hidden;
            height: 100dvh;
            /* Safe Areas für Notch / Home-Indicator */
            padding-top:    env(safe-area-inset-top, 0px);
            padding-bottom: env(safe-area-inset-bottom, 0px);
            padding-left:   env(safe-area-inset-left, 0px);
            padding-right:  env(safe-area-inset-right, 0px);
        }

        #app {
            display: flex;
            flex-direction: column;
            height: 100dvh;
        }

        /* Alle Touch-Targets mindestens 44px (Apple HIG) */
        button, .touchable {
            min-height: 44px;
            min-width: 44px;
            cursor: pointer;
        }

        /* Smooth Scrolling auf iOS */
        .scroll-view {
            overflow-y: auto;
            -webkit-overflow-scrolling: touch;
            overscroll-behavior: contain;
        }

        /* Text nicht versehentlich selektierbar */
        .no-select {
            user-select: none;
            -webkit-user-select: none;
        }
    </style>
</head>
<body>
    <div id="app"></div>
    <script type="module" src="/static/mobile/app.js"></script>
    <script>
        // Service Worker registrieren
        if ('serviceWorker' in navigator) {
            navigator.serviceWorker.register('/sw.js', { scope: '/mobile' })
                .then(reg => console.log('SW registriert:', reg.scope))
                .catch(err => console.warn('SW Fehler:', err));
        }
    </script>
</body>
</html>
```

1.2 app.js – Vue Root & Navigation

Datei: `static/mobile/app.js` (NEU)

Hinweise für KI:

- Kein Vue Router – Navigation über `ref` und `v-if`

(hält die Bundle-Size minimal)

- `initStore()` aus `store.js` ist identisch zur Desktop-App
- `globalStore` ist reaktiv – alle Views updaten automatisch
- Bottom-Navigation mit Badge für aktive Alarme
- `selectedDeviceId` steuert ob Overview oder Detail sichtbar ist

```
// static/mobile/app.js
import { globalStore, initStore } from '/static/js/store.js';
import OverviewView    from '/static/mobile/views/overview.js';
import DeviceDetailView from '/static/mobile/views/device_detail.js';
import AlarmsView      from '/static/mobile/views/alarms.js';
import ConnectionBanner from
'/static/mobile/components/connection_banner.js';

const { createApp, ref, computed, onMounted } = Vue;

createApp({
    components: {
        OverviewView, DeviceDetailView, AlarmsView, ConnectionBanner
    },

    setup() {
        // Navigation State
        const activeTab        = ref('overview');
        const selectedDeviceId = ref(null);

        // Gerätedetail öffnen
        const openDevice = (deviceId) => {
            selectedDeviceId.value = deviceId;
        };

        // Zurück zur Übersicht
        const goBack = () => {
            selectedDeviceId.value = null;
        };

        // Alarm-Badge Zähler
        const activeAlarmCount = computed(() => {
            let count = 0;
            Object.values(globalStore.devices).forEach(dev => {
                (dev.state || []).forEach(s => {
                    if (s.alarm_state && s.alarm_state !== 'NORMAL')
```

```
count++;
                });
            });
            return count;
        });

        // Tab wechseln (setzt auch Device-Detail zurück)
        const switchTab = (tab) => {
            activeTab.value = tab;
            selectedDeviceId.value = null;
        };

        // Globaler Notaus
        const emergencyStop = async () => {
            if (confirm('⚠ NOTAUS\nAlle Hardware-Ausgänge abschalten?')) {
                try {
                    await fetch('/api/emergency', { method: 'POST' });
                } catch (e) {
                    alert('Fehler: ' + e.message);
                }
            }
        };

        // Connectivity
        const isOnline = ref(navigator.onLine);
        window.addEventListener('online',  () => isOnline.value = true);
        window.addEventListener('offline', () => isOnline.value = false);

        onMounted(() => {
            initStore(); // ← Gleicher Store wie Desktop
        });

        // Bottom Navigation Tabs
        const tabs = [
            { id: 'overview', icon: 'view-dashboard', label: 'Geräte' },
            { id: 'alarms',   icon: 'bell-alert',     label: 'Alarme'  },
        ];

        return {
            store: globalStore,
            activeTab, selectedDeviceId, tabs, activeAlarmCount,
            isOnline, openDevice, goBack, switchTab, emergencyStop
        };
    },

    template: `
    <div id="app" class="no-select">

        <!-- HEADER -->
        <header style="
            display: flex; align-items: center;
```

```
            justify-content: space-between;
            padding: 10px 15px;
            background: #1c1c1f;
            border-bottom: 1px solid #3f3f46;
            flex-shrink: 0; z-index: 100;">

            <!-- Zurück-Button oder Logo -->
            <div style="display: flex; align-items: center; gap: 10px; min-
width: 80px;">
                <button v-if="selectedDeviceId"
                    @click="goBack"
                    style="background: transparent; border: none;
                           color: #0ea5e9; font-size: 22px; padding: 0;
                           display: flex; align-items: center;">
                    <i class="mdi mdi-arrow-left"></i>
                </button>
                <span v-else style="font-weight: bold; color: #0ea5e9; font-
size: 15px;">
                    ionpy <span style="opacity:0.5; font-
size:10px;">MOBILE</span>
                </span>
            </div>

            <!-- Titel bei Detail-View -->
            <div v-if="selectedDeviceId" style="
                font-size: 13px; font-weight: bold; color: #e4e4e7;
                overflow: hidden; text-overflow: ellipsis; white-space:
nowrap;
                max-width: 150px;">
                {{ store.devices[selectedDeviceId]?.meta?.alias ||
selectedDeviceId }}
            </div>

            <!-- Rechts: Status + Notaus -->
            <div style="display: flex; align-items: center; gap: 12px; min-
width: 80px; justify-content: flex-end;">
                <!-- WS Indikator -->
                <div :style="{
                    width: '8px', height: '8px', borderRadius: '50%',
                    background: store.wsConnected ? '#10b981' : '#ef4444',
                    boxShadow: store.wsConnected ? '0 0 6px #10b981' :
'none',
                    flex: 'none'
                }" :title="store.wsConnected ? 'Verbunden' :
'Getrennt'"></div>

                <!-- NOTAUS – immer sichtbar -->
                <button @click="emergencyStop" style="
                    background: #7f1d1d; color: #fca5a5;
                    border: 1px solid #991b1b;
                    padding: 6px 10px; border-radius: 6px;
```

```
                font-weight: bold; font-size: 11px;
                white-space: nowrap;">
                ⚠ NOTAUS
            </button>
        </div>
    </header>

    <!-- OFFLINE BANNER -->
    <connection-banner
        :ws-connected="store.wsConnected"
        :is-online="isOnline">
    </connection-banner>

    <!-- MAIN CONTENT -->
    <main class="scroll-view" style="flex: 1; overflow-y: auto;">
        <overview-view
            v-if="activeTab === 'overview' && !selectedDeviceId"
            @open-device="openDevice">
        </overview-view>

        <device-detail-view
            v-else-if="selectedDeviceId"
            :device-id="selectedDeviceId">
        </device-detail-view>

        <alarms-view
            v-else-if="activeTab === 'alarms'">
        </alarms-view>
    </main>

    <!-- BOTTOM NAVIGATION -->
    <nav style="
        display: flex;
        background: #1c1c1f;
        border-top: 1px solid #3f3f46;
        flex-shrink: 0;">

        <button v-for="tab in tabs" :key="tab.id"
            @click="switchTab(tab.id)"
            :style="{
                flex: 1,
                display: 'flex', flexDirection: 'column',
                alignItems: 'center', justifyContent: 'center',
                padding: '8px 5px',
                border: 'none',
                background: 'transparent',
                color: activeTab === tab.id && !selectedDeviceId
                    ? '#0ea5e9' : '#71717a',
                gap: '3px',
                position: 'relative',
                transition: 'color 0.15s'
```

```
                }">

                <i :class="'mdi mdi-' + tab.icon"
                   style="font-size: 24px;"></i>
                <span style="font-size: 10px; font-weight: 500;">
                    {{ tab.label }}
                </span>

                <!-- Alarm Badge -->
                <span v-if="tab.id === 'alarms' && activeAlarmCount > 0"
                    style="
                        position: absolute; top: 5px;
                        right: calc(50% - 20px);
                        background: #ef4444; color: white;
                        border-radius: 10px; font-size: 9px;
                        font-weight: bold; padding: 1px 5px;
                        min-width: 16px; text-align: center;
                        line-height: 1.4;">
                    {{ activeAlarmCount > 99 ? '99+' : activeAlarmCount }}
                </span>
            </button>
        </nav>
    </div>
    `
}).mount('#app');
```

PHASE 2: Core Views

(~3 Tage)

2.1 connection_banner.js – Verbindungsstatus

Datei: `static/mobile/components/connection_banner.js` (NEU)
Aufwand: 0.5 Tage

Hinweis für KI: Zeigt einen farbigen Banner wenn WS getrennt oder kein Netzwerk vorhanden. Verschwindet automatisch wenn Verbindung wiederhergestellt.

```
export default {
    props: {
        wsConnected: { type: Boolean, default: false },
        isOnline:    { type: Boolean, default: true  }
    },
    computed: {
        banner() {
```

```
            if (!this.isOnline) return {
                show: true,
                color: '#7f1d1d',
                icon:  'wifi-off',
                text:  'Kein Netzwerk – Live-Daten nicht verfügbar'
            };
            if (!this.wsConnected) return {
                show: true,
                color: '#78350f',
                icon:  'lan-disconnect',
                text:  'Verbindung getrennt – Stelle Verbindung wieder
her...'
            };
            return { show: false };
        }
    },
    template: `
    <div v-if="banner.show"
        :style="{
            background: banner.color,
            padding: '8px 15px',
            display: 'flex', alignItems: 'center', gap: '8px',
            fontSize: '12px', color: '#fca5a5',
            flexShrink: 0
        }">
        <i :class="'mdi mdi-' + banner.icon"></i>
        <span>{{ banner.text }}</span>
    </div>
    `
};
```

2.2 overview.js – Geräteübersicht

Datei: `static/mobile/views/overview.js` (NEU)
Aufwand: 1 Tag

Hinweis für KI:

- Responsiv: 1 Spalte auf Phone, 2 Spalten auf Tablet (>768px)
- Key-Metrics: Die ersten 3 Numeric-Entities nach `ui.order` sortiert
- Touch-Feedback: `pointerdown / pointerup` für sofortiges visuelles Feedback
- Alarm-Farben direkt in den Metric-Werten anzeigen

```
import { globalStore } from '/static/js/store.js';
import DeviceCard from '/static/mobile/components/device_card.js';

export default {
    components: { DeviceCard },
    emits: ['open-device'],
```

```
    setup(_, { emit }) {
        const { computed, ref } = Vue;

        // Responsiv
        const isTablet = ref(window.innerWidth >= 768);
        window.addEventListener('resize', () => {
            isTablet.value = window.innerWidth >= 768;
        });

        const devices = computed(() => {
            return Object.values(globalStore.devices)
                .map(dev => {
                    const stateEnt = dev.state?.find(s => s.sensor ===
'state');
                    const status   = stateEnt?.value ?? 'OFFLINE';
                    const statusColor = {
                        ONLINE:     '#10b981',
                        ERROR:      '#ef4444',
                        CONNECTING: '#f59e0b',
                        RECOVERING: '#f59e0b'
                    }[status] ?? '#71717a';

                    // Top 3 Messwerte für die Karte
                    const keyMetrics = (dev.meta?.entities ?? [])
                        .filter(e =>
                            e.ui?.ui_type === 'number' &&
                            e.mode !== 'WRITE_ONLY' &&
                            e.channel !== 'System'
                        )
                        .sort((a, b) => (a.ui?.order ?? 99) - (b.ui?.order
?? 99))
                        .slice(0, 3)
                        .map(e => {
                            const s = dev.state?.find(x => x.sensor ===
e.id);
                            return {
                                name:  e.name,
                                unit:  e.unit ?? '',
                                value: s?.value != null
                                    ? Number(s.value).toFixed(e.accuracy ??
1)
                                    : '--',
                                alarm: s?.alarm_state ?? 'NORMAL'
                            };
                        });

                    return {
                        id:         dev.meta.device_id,
                        name:       dev.meta.alias || dev.meta.device_id,
                        icon:       dev.meta.icon || 'developer-board',
                        status,
```

```
                        statusColor,
                        keyMetrics
                    };
                })
                .sort((a, b) => a.name.localeCompare(b.name));
        });

        return { devices, isTablet, emit };
    },

    template: `
    <div :style="{
        padding: '12px',
        display: 'grid',
        gridTemplateColumns: isTablet ? 'repeat(2, 1fr)' : '1fr',
        gap: '10px'
    }">
        <device-card
            v-for="dev in devices"
            :key="dev.id"
            :device="dev"
            @click="$emit('open-device', dev.id)">
        </device-card>

        <div v-if="devices.length === 0"
            style="grid-column: 1/-1; text-align: center;
                   padding: 60px 20px; color: #52525b;">
            <i class="mdi mdi-devices"
               style="font-size: 56px; display: block;
                      margin-bottom: 15px; opacity: 0.3;"></i>
            <div style="font-size: 14px;">Keine Geräte verfügbar</div>
            <div style="font-size: 12px; margin-top: 5px; color: #3f3f46;">
                Warte auf Verbindung...
            </div>
        </div>
    </div>
    `
};
```

2.3 device_card.js – Gerätekarte

Datei: `static/mobile/components/device_card.js` (NEU)
Aufwand: 0.5 Tage

```
export default {
    props: {
        device: { type: Object, required: true }
    },
```

```
    setup(props) {
        const { ref } = Vue;
        const pressed = ref(false);

        const alarmColor = (alarm) => {
            if (alarm === 'HIHI' || alarm === 'LOLO') return '#ef4444';
            if (alarm === 'HI'   || alarm === 'LO')   return '#f59e0b';
            return '#ffffff';
        };

        return { pressed, alarmColor };
    },

    template: `
    <div
        @pointerdown="pressed = true"
        @pointerup="pressed = false"
        @pointerleave="pressed = false"
        :style="{
            background: pressed ? '#3f3f46' : '#27272a',
            border: '1px solid',
            borderColor: device.status === 'ONLINE' ? '#3f3f46' : '#27272a',
            borderRadius: '12px',
            padding: '14px',
            cursor: 'pointer',
            transition: 'background 0.1s, transform 0.1s',
            transform: pressed ? 'scale(0.98)' : 'scale(1)',
            userSelect: 'none'
        }">

        <!-- Header: Name + Status -->
        <div style="display: flex; justify-content: space-between;
                    align-items: center; margin-bottom: 12px;">
            <div style="display: flex; align-items: center; gap: 8px;
                        overflow: hidden;">
                <i :class="'mdi mdi-' + device.icon"
                   style="font-size: 18px; color: #71717a; flex-shrink:
0;"></i>
                <span style="font-weight: bold; font-size: 14px;
                             white-space: nowrap; overflow: hidden;
                             text-overflow: ellipsis;">
                    {{ device.name }}
                </span>
            </div>
            <div style="display: flex; align-items: center;
                        gap: 5px; flex-shrink: 0; margin-left: 8px;">
                <div :style="{
                    width: '8px', height: '8px', borderRadius: '50%',
                    background: device.statusColor,
                    boxShadow: device.status === 'ONLINE'
                        ? '0 0 6px ' + device.statusColor : 'none'
```

```
                }"></div>
                <span style="font-size: 10px; color: #71717a;">
                    {{ device.status }}
                </span>
            </div>
        </div>

        <!-- Key Metrics -->
        <div v-if="device.keyMetrics.length > 0"
            style="display: flex; gap: 6px;">
            <div v-for="m in device.keyMetrics" :key="m.name"
                style="flex: 1; background: #18181b; border-radius: 8px;
                       padding: 8px 6px; text-align: center; min-width: 0;">
                <div :style="{
                    fontFamily: 'var(--font-mono)',
                    fontSize: '16px', fontWeight: 'bold',
                    color: alarmColor(m.alarm),
                    lineHeight: 1
                }">{{ m.value }}</div>
                <div style="font-size: 9px; color: #52525b;
                            margin-top: 4px; line-height: 1.2;
                            white-space: nowrap; overflow: hidden;
                            text-overflow: ellipsis;">
                    {{ m.name }}<span v-if="m.unit"> [{{ m.unit }}]</span>
                </div>
            </div>
        </div>

        <div v-else style="font-size: 12px; color: #52525b;
                           font-style: italic; text-align: center;
                           padding: 8px 0;">
            Keine Messwerte
        </div>

        <!-- Chevron -->
        <div style="text-align: right; margin-top: 8px;">
            <i class="mdi mdi-chevron-right"
               style="color: #3f3f46; font-size: 18px;"></i>
        </div>
    </div>
    `
};
```

2.4 device_detail.js – Gerät Detail

Datei: `static/mobile/views/device_detail.js` (NEU)
Aufwand: 1.5 Tage

Hinweis für KI:

- Zeigt ALLE Entities des Geräts gruppiert
- Für `toggle` Entities mit READ_WRITE: EIN/AUS Button
- Für `number` Entities: Wert + Einheit + Alarm-Farbe
- `sendControl()` nutzt den bestehenden `POST /api/control` Endpoint
- Mini-Chart (uPlot) für die ersten numerischen Entities
- Tabs für Kanäle (Ch1, Ch2, System) falls vorhanden

```
import { globalStore } from '/static/js/store.js';
import MiniChart from '/static/mobile/components/mini_chart.js';

export default {
    components: { MiniChart },
    props: {
        deviceId: { type: String, required: true }
    },

    setup(props) {
        const { computed, ref } = Vue;

        const device = computed(() => globalStore.devices[props.deviceId]);

        const currentState = computed(() => {
            const s = device.value?.state?.find(s => s.sensor === 'state');
            return s?.value ?? 'OFFLINE';
        });

        const statusColor = computed(() => ({
            ONLINE:     '#10b981',
            ERROR:      '#ef4444',
            CONNECTING: '#f59e0b',
            RECOVERING: '#f59e0b'
        }[currentState.value] ?? '#71717a'));

        // Kanäle ermitteln
        const channels = computed(() => {
            if (!device.value?.meta?.entities) return ['Ch1'];
            const chs = [...new Set(
                device.value.meta.entities
                    .filter(e => e.mode !== 'WRITE_ONLY')
                    .map(e => e.channel || 'Ch1')
            )];
            return chs.sort((a, b) => {
                if (a.toLowerCase() === 'system') return 1;
                if (b.toLowerCase() === 'system') return -1;
                return a.localeCompare(b);
            });
        });

        const activeChannel = ref('');

        // Aktiven Kanal initialisieren
```

```
        const initChannel = () => {
            if (!activeChannel.value && channels.value.length > 0) {
                activeChannel.value = channels.value[0];
            }
        };

        // Entities für aktiven Kanal, gruppiert
        const groupedEntities = computed(() => {
            initChannel();
            if (!device.value?.meta?.entities) return [];

            const visible = device.value.meta.entities.filter(e =>
                (e.channel || 'Ch1') === activeChannel.value &&
                e.mode !== 'WRITE_ONLY' &&
                e.ui?.ui_type !== 'table' &&
                e.ui?.ui_type !== 'waveform' &&
                !e.ui?.hidden
            );

            const map = {};
            visible.forEach(e => {
                const g = e.ui?.group || 'Allgemein';
                if (!map[g]) map[g] = {
                    name: g, order: e.ui?.group_order ?? 99, entities: []
                };
                map[g].entities.push(e);
            });

            return Object.values(map)
                .sort((a, b) => a.order - b.order)
                .map(g => ({
                    ...g,
                    entities: g.entities.sort(
                        (a, b) => (a.ui?.order ?? 99) - (b.ui?.order ?? 99)
                    )
                }));
        });

        // Wert aus State holen
        const getValue = (entityId) => {
            const s = device.value?.state?.find(s => s.sensor === entityId);
            return s?.value ?? null;
        };

        const getAlarmState = (entityId) => {
            const s = device.value?.state?.find(s => s.sensor === entityId);
            return s?.alarm_state ?? 'NORMAL';
        };

        const formatValue = (entity) => {
            const val = getValue(entity.id);
```

```
            if (val === null) return '--';
            if (entity.ui?.ui_type === 'toggle') {
                return val ? 'AN' : 'AUS';
            }
            if (typeof val === 'number') {
                return Number(val).toFixed(entity.accuracy ?? 2);
            }
            return String(val);
        };

        const alarmColor = (entityId) => {
            const state = getAlarmState(entityId);
            if (state === 'HIHI' || state === 'LOLO') return '#ef4444';
            if (state === 'HI'   || state === 'LO')   return '#f59e0b';
            return '#ffffff';
        };

        const isTrue = (val) => {
            if (val === true || val === 1 || val === '1') return true;
            if (typeof val === 'string') {
                return ['on','true','an'].includes(val.toLowerCase());
            }
            return false;
        };

        // Steuern
        const sendControl = async (entityId, value) => {
            // Optimistic Update
            const s = device.value?.state?.find(s => s.sensor === entityId);
            if (s) s.value = value;

            await fetch('/api/control', {
                method:  'POST',
                headers: { 'Content-Type': 'application/json' },
                body:    JSON.stringify({
                    device_id: props.deviceId,
                    key:       entityId,
                    value:     value
                })
            });
        };

        const toggleConnection = async () => {
            const target = currentState.value === 'ONLINE'
                ? 'OFFLINE' : 'ONLINE';
            await sendControl('state', target);
        };

        // Erste numerische Entities für Mini-Charts
        const chartEntities = computed(() => {
            if (!device.value?.meta?.entities) return [];
```

```
            return device.value.meta.entities
                .filter(e =>
                    e.ui?.ui_type === 'number' &&
                    e.mode !== 'WRITE_ONLY' &&
                    e.channel !== 'System'
                )
                .sort((a, b) => (a.ui?.order ?? 99) - (b.ui?.order ?? 99))
                .slice(0, 2); // Max 2 Charts auf Mobile
        });

        return {
            device, currentState, statusColor, channels, activeChannel,
            groupedEntities, getValue, formatValue, alarmColor, isTrue,
            sendControl, toggleConnection, chartEntities
        };
    },

    template: `
    <div v-if="device" style="display: flex; flex-direction: column; min-
height: 100%;">

        <!-- Status Bar -->
        <div style="display: flex; justify-content: space-between;
                    align-items: center; padding: 10px 15px;
                    background: #1c1c1f; border-bottom: 1px solid #3f3f46;">

            <div style="display: flex; align-items: center; gap: 8px;">
                <div :style="{
                    width: '10px', height: '10px', borderRadius: '50%',
                    background: statusColor,
                    boxShadow: '0 0 8px ' + statusColor
                }"></div>
                <span style="font-size: 12px; color: #a1a1aa;">
                    {{ currentState }}
                </span>
            </div>

            <!-- Connect / Disconnect -->
            <button @click="toggleConnection" :style="{
                background: currentState === 'ONLINE' ? '#3f3f46' :
'#10b981',
                color: 'white', border: 'none',
                padding: '6px 14px', borderRadius: '6px',
                fontSize: '12px', fontWeight: 'bold'
            }">
                {{ currentState === 'ONLINE' ? '□ Trennen' : '▶ Verbinden'
}}
            </button>
        </div>

        <!-- Mini Charts -->
```

```
        <div v-if="chartEntities.length > 0"
            style="padding: 10px 15px; background: #18181b;
                   border-bottom: 1px solid #27272a;">
            <div :style="{
                display: 'grid',
                gridTemplateColumns: chartEntities.length > 1
                    ? 'repeat(2, 1fr)' : '1fr',
                gap: '10px'
            }">
                <div v-for="ent in chartEntities" :key="ent.id"
                    style="background: #27272a; border-radius: 8px;
                           padding: 10px;">
                    <div style="font-size: 10px; color: #71717a;
                                margin-bottom: 6px;">
                        {{ ent.name }}
                    </div>
                    <mini-chart
                        :device-id="deviceId"
                        :entity-id="ent.id"
                        :color="'#0ea5e9'"
                        :window-ms="60000">
                    </mini-chart>
                    <div style="font-family: var(--font-mono);
                                font-size: 18px; font-weight: bold;
                                color: #0ea5e9; margin-top: 6px; text-align:
right;">
                        {{ getValue(ent.id) !== null
                            ? Number(getValue(ent.id)).toFixed(ent.accuracy
?? 2)
                            : '--' }}
                        <span style="font-size: 11px; color: #71717a;">
                            {{ ent.unit }}
                        </span>
                    </div>
                </div>
            </div>
        </div>

        <!-- Kanal Tabs -->
        <div v-if="channels.length > 1"
            style="display: flex; overflow-x: auto; background: #1c1c1f;
                   border-bottom: 1px solid #3f3f46; padding: 0 10px;">
            <button v-for="ch in channels" :key="ch"
                @click="activeChannel = ch"
                :style="{
                    padding: '10px 14px', border: 'none',
                    background: 'transparent', whiteSpace: 'nowrap',
                    fontSize: '12px', fontWeight: 'bold',
                    color: activeChannel === ch ? '#fff' : '#71717a',
                    borderBottom: activeChannel === ch
                        ? '2px solid #0ea5e9' : '2px solid transparent',
```

```
                    cursor: 'pointer', flexShrink: 0
                }">
                {{ ch }}
            </button>
        </div>

        <!-- Entity Liste -->
        <div style="padding: 10px;">
            <div v-for="group in groupedEntities" :key="group.name"
                style="margin-bottom: 16px;">

                <!-- Gruppen-Header -->
                <div style="font-size: 10px; font-weight: bold;
                            color: #0ea5e9; text-transform: uppercase;
                            letter-spacing: 0.5px; margin-bottom: 6px;
                            padding-left: 2px;">
                    {{ group.name }}
                </div>

                <!-- Entities -->
                <div style="background: #27272a; border-radius: 10px;
                            overflow: hidden;">
                    <div v-for="(ent, idx) in group.entities" :key="ent.id"
                        :style="{
                            display: 'flex',
                            justifyContent: 'space-between',
                            alignItems: 'center',
                            padding: '12px 14px',
                            borderBottom: idx < group.entities.length - 1
                                ? '1px solid #3f3f46' : 'none'
                        }">

                        <!-- Label -->
                        <div style="flex: 1; overflow: hidden;">
                            <div style="font-size: 13px; color: #a1a1aa;
                                        white-space: nowrap; overflow:
hidden;
                                        text-overflow: ellipsis;">
                                {{ ent.name }}
                            </div>
                            <div v-if="ent.unit"
                                style="font-size: 10px; color: #52525b;
                                       font-family: var(--font-mono);">
                                [{{ ent.unit }}]
                            </div>
                        </div>

                        <!-- Wert / Control -->
                        <div style="margin-left: 12px; flex-shrink: 0;">

                            <!-- Toggle RW -->
```

```
                            <button
                                v-if="ent.ui?.ui_type === 'toggle' &&
ent.mode === 'READ_WRITE'"
                                @click="sendControl(ent.id,
!isTrue(getValue(ent.id)))"
                                :style="{
                                    background: isTrue(getValue(ent.id))
                                        ? '#10b981' : '#3f3f46',
                                    border: 'none', color: 'white',
                                    padding: '8px 20px', borderRadius:
'6px',
                                    fontWeight: 'bold', fontSize: '13px',
                                    minWidth: '70px'
                                }">
                                {{ isTrue(getValue(ent.id)) ? 'AN' : 'AUS'
}}
                            </button>

                            <!-- Toggle RO (LED) -->
                            <div v-else-if="ent.ui?.ui_type === 'toggle'"
                                style="display: flex; align-items: center;
gap: 6px;">
                                <div :style="{
                                    width: '12px', height: '12px',
                                    borderRadius: '50%',
                                    background: isTrue(getValue(ent.id))
                                        ? '#10b981' : '#3f3f46',
                                    boxShadow: isTrue(getValue(ent.id))
                                        ? '0 0 6px #10b981' : 'none'
                                }"></div>
                                <span style="font-size: 12px; color:
#a1a1aa;">
                                    {{ isTrue(getValue(ent.id)) ? 'AN' :
'AUS' }}
                                </span>
                            </div>

                            <!-- Numerisch / Text -->
                            <div v-else
                                style="text-align: right;">
                                <span :style="{
                                    fontFamily: 'var(--font-mono)',
                                    fontSize: '18px', fontWeight: 'bold',
                                    color: alarmColor(ent.id)
                                }">
                                    {{ formatValue(ent) }}
                                </span>
                            </div>
                        </div>
                    </div>
                </div>
```

```
            </div>
        </div>
    </div>

    <div v-else style="display: flex; align-items: center;
                       justify-content: center; height: 200px;
                       color: #52525b; font-style: italic;">
        Gerät nicht gefunden
    </div>
    `
};
```

2.5 mini_chart.js – uPlot Sparkline

Datei: `static/mobile/components/mini_chart.js` (NEU)
Aufwand: 0.5 Tage

Hinweis für KI:

- Hört auf `ws-message` CustomEvents (gleicher Mechanismus wie Desktop)
- ResizeObserver für responsive Breite – kein fixer Wert
- Kein Cursor, keine Legende – nur die Kurve
- `onUnmounted` muss EventListener UND ResizeObserver trennen

```
export default {
    props: {
        deviceId:  { type: String,  required: true },
        entityId:  { type: String,  required: true },
        color:     { type: String,  default: '#0ea5e9' },
        windowMs:  { type: Number,  default: 60000 },
        height:    { type: Number,  default: 50 }
    },

    setup(props) {
        const { ref, onMounted, onUnmounted } = Vue;
        const chartEl = ref(null);

        let uplot = null;
        let resizeObserver = null;
        const data = [[], []];

        const buildChart = (width) => {
            if (!chartEl.value || width <= 0) return;
            if (uplot) { uplot.destroy(); uplot = null; }

            const opts = {
                width,
                height:  props.height,
                legend:  { show: false },
                cursor:  { show: false },
```

```
            select:  { show: false },
            scales:  { x: { time: true }, y: { auto: true } },
            axes:    [{ show: false }, { show: false }],
            series:  [
                {},
                {
                    stroke: props.color,
                    width:  2,
                    fill:   props.color + '20'
                }
            ],
            padding: [2, 0, 2, 0]
        };

        uplot = new uPlot(opts, data, chartEl.value);
    };

    const handleWsMessage = (e) => {
        try {
            const sample = JSON.parse(e.detail);
            if (sample.device_id !== props.deviceId) return;
            if (sample.entity_id !== props.entityId)  return;
            if (typeof sample.value !== 'number')      return;
            if (sample.type === 'waveform')            return;

            const now    = Date.now() / 1000;
            const cutoff = now - (props.windowMs / 1000);

            data[0].push(now);
            data[1].push(sample.value);

            // Alte Werte entfernen
            while (data[0].length > 0 && data[0][0] < cutoff) {
                data[0].shift();
                data[1].shift();
            }

            if (uplot && data[0].length > 1) {
                uplot.setData(data, true);
            }
        } catch { /* Ignore parse errors */ }
    };

    onMounted(async () => {
        await Vue.nextTick();
        if (!chartEl.value) return;

        resizeObserver = new ResizeObserver(entries => {
            const w = Math.floor(entries[0].contentRect.width);
            if (!uplot && w > 0) {
                buildChart(w);
```

```
                } else if (uplot && w > 0) {
                    uplot.setSize({ width: w, height: props.height });
                }
            });
            resizeObserver.observe(chartEl.value);

            window.addEventListener('ws-message', handleWsMessage);
        });

        onUnmounted(() => {
            window.removeEventListener('ws-message', handleWsMessage);
            if (resizeObserver) resizeObserver.disconnect();
            if (uplot) uplot.destroy();
        });

        return { chartEl };
    },

    template: `<div ref="chartEl" style="width: 100%;"></div>`
};
```

2.6 alarms.js – Alarm Feed

Datei: `static/mobile/views/alarms.js` (NEU)
Aufwand: 0.5 Tage

```
import { globalStore } from '/static/js/store.js';

export default {
    setup() {
        const { computed } = Vue;

        const allAlarms = computed(() => {
            const result = [];
            Object.values(globalStore.devices).forEach(dev => {
                (dev.state || []).forEach(s => {
                    if (!s.alarm_state || s.alarm_state === 'NORMAL')
return;

                    const ent = dev.meta?.entities?.find(e => e.id ===
s.sensor);
                    result.push({
                        deviceId:   dev.meta.device_id,
                        deviceName: dev.meta.alias || dev.meta.device_id,
                        entityId:   s.sensor,
                        entityName: ent?.name || s.sensor,
                        value:      s.value,
                        unit:       ent?.unit || '',
                        alarm:      s.alarm_state,
```

```
                        timestamp:  s.timestamp
                    });
                });
            });
            // Kritische zuerst
            const priority = { HIHI: 0, LOLO: 1, HI: 2, LO: 3 };
            return result.sort((a, b) =>
                (priority[a.alarm] ?? 9) - (priority[b.alarm] ?? 9)
            );
        });

        const alarmStyle = (alarm) => {
            const crit = alarm === 'HIHI' || alarm === 'LOLO';
            return {
                background:   crit ? '#450a0a' : '#431407',
                border:       `1px solid ${crit ? '#991b1b' : '#78350f'}`,
                color:        crit ? '#fca5a5' : '#fed7aa',
                borderRadius: '10px',
                padding:      '12px 14px',
                marginBottom: '8px'
            };
        };

        const formatTime = (ts) => {
            if (!ts) return '--';
            const d = new Date(ts * 1000);
            return d.toLocaleTimeString('de-DE', { hour12: false }) +
                   '.' + String(d.getMilliseconds()).padStart(3, '0');
        };

        return { allAlarms, alarmStyle, formatTime };
    },

    template: `
    <div style="padding: 12px;">
        <div v-if="allAlarms.length === 0"
            style="text-align: center; padding: 60px 20px; color: #52525b;">
            <i class="mdi mdi-check-circle-outline"
               style="font-size: 56px; display: block;
                      margin-bottom: 15px; color: #10b981; opacity:
0.5;"></i>
            <div style="font-size: 14px;">Keine aktiven Alarme</div>
        </div>

        <div v-for="alarm in allAlarms" :key="alarm.deviceId +
alarm.entityId"
            :style="alarmStyle(alarm.alarm)">
            <div style="display: flex; justify-content: space-between;
                        align-items: flex-start;">
                <div>
                    <div style="font-weight: bold; font-size: 13px;
```

```
                                margin-bottom: 4px;">
                            {{ alarm.deviceName }} › {{ alarm.entityName }}
                        </div>
                        <div style="font-size: 12px; opacity: 0.8;">
                            {{ alarm.alarm }} —
                            <span style="font-family: var(--font-mono);
                                         font-weight: bold;">
                                {{ alarm.value }} {{ alarm.unit }}
                            </span>
                        </div>
                    </div>
                    <div style="font-size: 10px; opacity: 0.6;
                                font-family: var(--font-mono);
                                flex-shrink: 0; margin-left: 10px;">
                        {{ formatTime(alarm.timestamp) }}
                    </div>
                </div>
            </div>
        </div>
    `
};
```

PHASE 3: PWA Install & Offline

(~0.5 Tage – fast geschenkt)

Nach Phase 2 ist die App bereits **voll funktionsfähig** im Browser. Phase 3 macht sie installierbar.

3.1 iOS Homescreen

Auf iPhone/iPad: **Safari → Teilen-Button → Zum Home-Bildschirm**

Dafür müssen die `<meta>` Tags in `mobile.html` korrekt gesetzt sein (bereits in Phase 1 enthalten).

3.2 Android Install Banner

Chrome auf Android zeigt automatisch einen Install-Banner wenn:

- Manifest vorhanden
- Service Worker registriert
- HTTPS oder localhost

Kein zusätzlicher Code nötig.

3.3 HTTPS für produktiven Einsatz

Hinweis für KI: Für den Einsatz im internen Netz (nicht localhost) braucht PWA auf iOS zwingend HTTPS. Optionen:

- **nginx Reverse Proxy** mit self-signed Cert
- **Caddy** (automatisches HTTPS, einfachste Option)
- **mkcert** für lokale CA im Netzwerk

```
# Caddy Beispiel (Caddyfile):
# ionpy.local {
#     reverse_proxy localhost:8000
# }
# → Caddy holt automatisch ein lokales Cert
```

PHASE 4: Erweiterungen (Nice-to-have)

4.1 Wake Lock – Bildschirm bleibt an

Aufwand: 30 Minuten
Bestes Aufwand/Nutzen Verhältnis im ganzen Projekt

```
// In app.js setup() ergänzen:
let wakeLock = null;

const enableWakeLock = async () => {
    if (!('wakeLock' in navigator)) return;
    try {
        wakeLock = await navigator.wakeLock.request('screen');
    } catch (e) {
        console.warn('Wake Lock nicht verfügbar:', e);
    }
};

// Neu anfordern wenn Tab wieder sichtbar wird
document.addEventListener('visibilitychange', async () => {
    if (document.visibilityState === 'visible') {
        await enableWakeLock();
    }
});

onMounted(() => {
    initStore();
    enableWakeLock(); // Sofort aktivieren
```

```
});
```

4.2 Vibrations-Alarm

Aufwand: 30 Minuten

```
// In alarms.js oder store.js – bei neuem Alarm auslösen:
const vibrateOnNewAlarm = (alarmState) => {
    if (!('vibrate' in navigator)) return; // iOS unterstützt das nicht
    if (alarmState === 'HIHI' || alarmState === 'LOLO') {
        navigator.vibrate([500, 100, 500, 100, 500]); // Kritisch
    } else if (alarmState === 'HI' || alarmState === 'LO') {
        navigator.vibrate(200); // Warnung
    }
};
```

4.3 Vollbild-Einzelwert (Distanz-Monitoring)

Aufwand: 0.5 Tage
Usecase: Tablet liegt auf dem Tisch – Wert lesbar aus 3m Entfernung

```
// static/mobile/views/sensor_fullscreen.js (NEU)
// Wird durch langen Tap auf einen Sensor-Tile geöffnet
// Schließt durch Tap irgendwo

// clamp(min, preferred, max) sorgt für skalierenden Text
// fontSize: 'clamp(80px, 25vw, 220px)'
```

4.4 Swipe-Gesten

Aufwand: 0.5 Tage

```
// Swipe von links → Detail schließen (wie native App)
// In device_detail.js:
const setupSwipeBack = (onSwipe) => {
    let startX = 0;
    const onTouchStart = e => { startX = e.touches[0].clientX; };
    const onTouchEnd = e => {
        const delta = e.changedTouches[0].clientX - startX;
        if (delta > 80 && startX < 50) onSwipe();
    };
    return { onTouchStart, onTouchEnd };
};
```

4.5 QR-Code Scanner

Aufwand: 1 Tag
Usecase: QR-Code klebt am Gerät → Handy draufhalten → Detail öffnet sich sofort

```
// Benötigt jsQR Library (oder BarcodeDetector API auf Android Chrome)
// QR-Format: "ionpy://device/DPS5005_1"
// → Parsen → openDevice(deviceId) aufrufen
```

4.6 Handy als Sensor-Device

Aufwand: 2-3 Tage
Beschreibung: Handy sendet eigene Sensordaten als ionpy-Device ans Backend. Separat dokumentiert in der Kamera/Sensor-Roadmap.

4.7 Push Notifications

Aufwand: 2 Tage
Voraussetzung: HTTPS + Service Worker (bereits vorhanden)

Alarm-Benachrichtigung auch wenn App geschlossen ist. Braucht Backend-seitigen Web-Push Service (`pywebpush` Library).

4.8 Dark/Light Theme Toggle

Aufwand: 0.5 Tage
Systemtheme via `prefers-color-scheme` Media Query + manueller Toggle. Theme in `localStorage` speichern.

4.9 Capacitor Wrapper (echter App Store)

Aufwand: 2-3 Tage Setup
Wenn die PWA stabil läuft kann sie mit Capacitor in eine native iOS/Android App verpackt werden. Der Vue-Code bleibt 1:1 identisch.

```
npm install @capacitor/core @capacitor/cli
npx cap init ionpy com.ionpy.mobile
npx cap add ios
npx cap add android
npx cap copy
npx cap open ios    # Öffnet Xcode
npx cap open android # Öffnet Android Studio
```

Gesamtübersicht Zeitplan PWA

Phase	Inhalt	Tage	Ergebnis
0	Vorbereitung + SW	0.5	Dateistruktur steht
1	Entry Point + Navigation	1.0	App startet im Browser
2	Core Views	3.0	Vollständig nutzbare App
3	PWA Install	0.5	Auf Homescreen installierbar
MVP		**~5 Tage**	**Produktionsreif**
4.1	Wake Lock	0.5h	Bildschirm bleibt an
4.2	Vibration	0.5h	Alarm-Feedback
4.3	Vollbild-Sensor	0.5	Distanz-Monitoring
4.4	Swipe-Gesten	0.5	Nativer App-Feel
4.5	QR-Scanner	1.0	Geräte-Shortcut
4.6	Handy als Sensor	3.0	Rückkanal aktiv
4.7	Push Notifications	2.0	Alarme im Hintergrund

Unabhängigkeit: Die PWA hat **keine Abhängigkeiten** vom Wizard-Backlog. Kann sofort parallel entwickelt werden.

Ende PWA Roadmap
Code-Beispiele sind Implementierungshinweise – kein fertiger Produktionscode