Smart Inspection

سند معماری سامانه ردیابی نیروهای میدانی

نسخه فارسی — معماری، حریم خصوصی و جریان داده‌های مکانی

> هدف سند: شرح ساختار فنی سامانه، کنترل‌های حریم خصوصی، منطق شیفت و نحوه‌ی ذخیره‌سازی و نمایش موقعیت نیروهای میدانی.

معرفی

Smart Inspection یک سامانه‌ی ردیابی موقعیت برای نیروهای میدانی است. این سامانه به سرپرست‌ها اجازه می‌دهد فقط در ساعات کاری، موقعیت زنده و تاریخچه‌ی مسیر کارکنان را مشاهده کنند.

اصل بنیادین سامانه این است: خارج از بازه‌ی شیفت، هیچ داده‌ی مکانی از هیچ کارمندی ذخیره یا نمایش داده نمی‌شود.

این قانون هسته‌ی معماری، منطق کسب‌وکار و طراحی امنیتی سامانه را تشکیل می‌دهد.

۱. معماری کلی

پروژه از سه بخش مستقل تشکیل شده است:

- /panel — پنل مدیریت و مشاهده‌ی موقعیت‌ها، مبتنی بر Nuxt 4، Vue 3 و TypeScript
- /api — بک‌اند Laravel و محل اجرای تمام منطق‌های کسب‌وکار
- /app — اپلیکیشن Flutter برای Android

این سه بخش فقط از طریق API مبتنی بر HTTP و WebSocket با یکدیگر ارتباط دارند و هیچ‌کدام دسترسی مستقیم به دیتابیس بخش دیگر ندارند.

۲. بک‌اند

بک‌اند در مسیر /api قرار دارد و تمام منطق کسب‌وکار سیستم در این بخش متمرکز است.

فناوری‌های اصلی

- Laravel 13
- PHP 8.2
- PostgreSQL
- PostGIS
- Redis
- Laravel Reverb
- Laravel Sanctum

برای ذخیره‌سازی داده‌های مکانی از PostgreSQL و افزونه‌ی PostGIS استفاده می‌شود. جدول location_points با توجه به رشد احتمالی حجم داده‌ها، به‌صورت ماهانه پارتیشن‌بندی می‌شود.

۳. احراز هویت

احراز هویت با Laravel Sanctum پیاده‌سازی شده است. پنل مدیریت از کوکی نشست و اپلیکیشن موبایل از توکن API استفاده می‌کند. هر دو مدل در یک سامانه‌ی احراز هویت و Guard مشترک مدیریت می‌شوند.

۴. پنل مدیریت

پنل مدیریت در مسیر /panel قرار دارد.

فناوری‌های پنل

- Nuxt 4
- Vue 3
- TypeScript
- Tailwind CSS
- MapLibre GL
- Laravel Echo

نقشه با MapLibre GL و کاشی‌های استاندارد OpenStreetMap نمایش داده می‌شود. ارتباط بلادرنگ نیز به‌وسیله‌ی Laravel Echo و Laravel Reverb برقرار است.

۵. اپلیکیشن موبایل

اپلیکیشن موبایل در مسیر app/ قرار دارد و با Flutter توسعه داده شده است.

ویژگی‌های اصلی

- سرویس پیش‌زمینه (Foreground Service)
- نمایش اعلان پایدار هنگام فعال بودن ردیابی
- دریافت موقعیت GPS
- صف محلی با SQLite
- پشتیبانی از حالت آفلاین
- فایل APK یکپارچه
- پشتیبانی از ARM64 و ARMv7
- امضای APK با کلید اختصاصی

اپلیکیشن بدون وابستگی به Google Play Store قابل توزیع است.

۶. هسته‌ی منطق کسب‌وکار

مهم‌ترین اصل در معماری Smart Inspection، متمرکز بودن منطق ردیابی است.

ShiftWindowResolver

این سرویس تنها مرجع تصمیم‌گیری درباره‌ی وضعیت شیفت یک کارمند است. هر بخش از سامانه که باید تشخیص دهد کارمند در حال حاضر داخل شیفت کاری قرار دارد یا نه، فقط از همین Resolver استفاده می‌کند.

TrackingGate

پیش از ذخیره‌ی هر نقطه‌ی GPS، درخواست باید از TrackingGate عبور کند. TrackingGate وضعیت شیفت را از ShiftWindowResolver دریافت می‌کند.

```
داخل شیفت ← ذخیره‌سازی
خارج از شیفت یا وضعیت نامشخص ← رد درخواست
```

اگر وضعیت شیفت خارج از شیفت یا نامشخص باشد، نقطه‌ی مکانی اصلاً ذخیره نمی‌شود؛ نه حذف نرم و نه نشانه‌گذاری. رفتار پیش‌فرض سامانه، رد کردن درخواست است.

۷. جریان داده‌های مکانی

```
GPS
↓
بررسی هر ۱۰ ثانیه
↓
ارسال هر ۱۵ ثانیه، مشروط به حداقل ۲۰ متر جابه‌جایی
↓
صف محلی SQLite
↓
ارسال به API
↓
TrackingGate
↓
ShiftWindowResolver
├── داخل شیفت ← PostgreSQL / location_points
└── خارج از شیفت یا نامشخص ← رد درخواست
```

این معماری باعث می‌شود حتی در صورت ارسال داده از اپلیکیشن خارج از ساعت کاری، بک‌اند همچنان آخرین خط دفاعی را اجرا کند.

۸. موقعیت زنده و تاریخچه‌ی موقعیت

تاریخچه‌ی موقعیت

برای نمایش تاریخچه‌ی مسیر، داده‌ها در location_points ذخیره می‌شوند و تنها پس از عبور از TrackingGate وارد فضای ذخیره‌سازی خواهند شد.

پینگ زنده

برای موقعیت زنده، یک جریان جداگانه و سبک‌تر وجود دارد. این پینگ‌ها در هر چرخه‌ی بررسی ارسال می‌شوند، روی دیسک ذخیره نمی‌شوند و فقط برای به‌روزرسانی نقشه‌ی زنده کاربرد دارند.

> پینگ زنده ← WebSocket ← نقشه‌ی سرپرست

۹. معماری بلادرنگ

> اپلیکیشن Flutter
> ↓ پینگ زنده
> Laravel API
> ↓
> Laravel Reverb
> WebSocket ↓
> پنل مدیریت
> ↓
> MapLibre

سرپرست می‌تواند موقعیت نیروهای فعال را با تأخیر بسیار کم روی نقشه مشاهده کند.

۱۰. حریم خصوصی در سطح طراحی

حریم خصوصی یک قابلیت جانبی نیست؛ بلکه در خود معماری پیاده‌سازی شده است.

اگر کارمند خارج از شیفت باشد، داده‌ی مکانی نباید وجود داشته باشد. در نتیجه سامانه به‌جای ذخیره‌ی داده و پنهان‌کردن آن در رابط کاربری، از ابتدا اجازه‌ی ذخیره‌سازی را نمی‌دهد.

> خارج از شیفت ← TrackingGate ← رد درخواست ← عدم ثبت در دیتابیس

۱۱. مجوزدهی

مجوزدهی بر پایه‌ی قابلیت‌ها طراحی شده است، نه فقط سلسله‌مراتب نقش‌ها.

> مدیریت شیفت‌ها ≠ مشاهده‌ی موقعیت کارکنان

برای نمونه، یک کاربر می‌تواند اجازه‌ی مدیریت شیفت‌ها را داشته باشد، اما اجازه‌ی مشاهده‌ی موقعیت مکانی کارکنان را نداشته باشد. هر قابلیت به‌صورت مستقل کنترل می‌شود.

۱۲. ثبت ممیزی دسترسی

هر بار که تاریخچه‌ی موقعیت یک کارمند مشاهده می‌شود، یک رکورد در access_audit_log ثبت خواهد شد.

- این گزارش فقط به‌صورت افزایشی ثبت می‌شود (Append-only).
- برای ثبت دسترسی به داده‌های حساس استفاده می‌شود.
- از طریق کد برنامه قابل حذف نیست.

بنابراین مشخص است چه شخصی و در چه زمانی به تاریخچه‌ی موقعیت دسترسی داشته است.

۱۳. یکپارچگی شیفت

تغییرات مربوط به شیفت نباید بتوانند تاریخچه‌ی گذشته را تغییر دهند. به همین دلیل effective_from نمی‌تواند در گذشته قرار بگیرد.

در نتیجه ایجاد یا تغییر یک شیفت جدید، قادر به بازنویسی قوانین مربوط به داده‌های مکانی گذشته نیست.

۱۴. ابطال فوری توکن

در صورت غیرفعال‌شدن حساب کاربری، تغییر رمز عبور یا دیگر رخدادهای امنیتی مرتبط، توکن‌های کاربر بلافاصله باطل می‌شوند. بنابراین نشست یا توکن قدیمی پس از تغییر وضعیت امنیتی حساب، قابل استفاده نخواهد بود.

۱۵. زیرساخت

کل سامانه برای مقیاس فعلی به‌صورت آگاهانه ساده نگه داشته شده است.

```
یک VPS
↓
Docker Compose
├— API
├— Panel
├— PostgreSQL
├— Redis
└— Reverb
```

ظرفیت هدف حدود ۵۰ کارمند و سقف طراحی فعلی حدود ۱۵۰ کارمند است. در این مقیاس، استفاده از Kubernetes یا معماری مایکروسرویس پیچیدگی غیرضروری ایجاد می‌کند؛ بنابراین راهکار به‌صورت بک‌اند یکپارچه با کلاینت‌های مستقل طراحی شده است.

۱۶. استقرار و انتشار

```
توسعه‌دهنده
↓
Git Push به شاخه main
↓
GitHub Actions
↓
SSH به VPS
↓
Docker Build
↓
Docker Compose
↓
به‌روزرسانی سرویس‌ها
```

هر Push به شاخه‌ی main می‌تواند فرایند انتشار را اجرا کند و نسخه‌ی جدید را روی VPS ساخته و مستقر کند.

۱۷. اصول معماری

- حریم خصوصی در سطح طراحی: موقعیت خارج از شیفت اصلاً ذخیره نمی‌شود.
- منبع واحد حقیقت: تمام تصمیم‌گیری‌های شیفت از ShiftWindowResolver عبور می‌کند.
- رد پیش‌فرض: اگر وضعیت شیفت مشخص نباشد، درخواست رد می‌شود.
- اعمال قوانین در بک‌اند: کنترل‌های امنیتی فقط در کلاینت اجرا نمی‌شوند.
- تفکیک مسئولیت‌ها: اپلیکیشن موبایل برای ردیابی، API برای منطق کسب‌وکار و پنل برای نمایش داده است.
- مجوزدهی مبتنی بر قابلیت: دسترسی‌ها با قابلیت‌های مستقل کنترل می‌شوند.
- قابلیت ممیزی: دسترسی به تاریخچه‌ی موقعیت قابل ثبت و بررسی است.
- زیرساخت ساده: در مقیاس فعلی، Docker Compose روی VPS نسبت به معماری توزیع‌شده ارجح است.

جمع‌بندی

Smart Inspection یک سامانه‌ی ردیابی موقعیت برای نیروی کار میدانی است که تمرکز آن فقط نمایش موقعیت نیست، بلکه کنترل دقیق زمان، دسترسی و چرخه‌ی عمر داده‌های مکانی را نیز پوشش می‌دهد.

مهم‌ترین ویژگی معماری این است: خارج از ساعات کاری، داده‌ی مکانی اصلاً ایجاد و ذخیره نمی‌شود.

این قانون با ترکیب ShiftWindowResolver، TrackingGate، رد پیش‌فرض، اعمال قوانین در بک‌اند، مجوزدهی مبتنی بر قابلیت و ثبت ممیزی دسترسی، در تمام لایه‌های سامانه اجرا می‌شود.

در نتیجه اپلیکیشن موبایل، پنل مدیریت و پردازش‌های پس‌زمینه همگی از یک قانون مرکزی برای تصمیم‌گیری درباره‌ی ردیابی استفاده می‌کنند و منطق‌های موازی یا متناقض برای شیفت شکل نمی‌گیرد.