جات (Route Optimizer)

أداة Backend تستقبل ملف إكسل فيه قائمة أماكن تسليم، وترتبها تلقائيًا في أفضل مسار زيارة (أقل مسافة ممكنة)، وترجع النتيجة كخريطة + PDF قابل للتحميل.

فكرة المشروع

د "مكان" (50-150 صف تقريبًا). المطلوب: تجاهل عمود المنتج تمامًا، عمود الأماكن بس، وترتيبها في مسار واحد منطقي يبدأ من موقع المندوب الحالي ويزور كل الأماكن بأقل مجهود ممكن — من غير الحاجة لأي ترتيب مسبق يدوي.

الاستاك المستخدم

الطبقة	التقنية
Backend	Node.js + Express
قاعدة البيانات	MongoDB (عبر Mongoose)
قراءة الإكسل	ExcelJS
تحويل العناوين لإحداثيات	Nominatim (OpenStreetMap) مجاني
ترتيب المسار	خوارزمية ذاتية: Nearest Neighbor + تحسين 2-opt
صورة الخريطة	Geoapify Static Maps API — مجاني (يحتاج API Key مجاني)
توليد PDF	pdfkit + خط عربي (Amiri) + arabic-reshaper

عتماد على APIs مدفوعة. كل الخدمات المستخدمة مجانية بالكامل، والوحيدة التي تتطلب تسجيل حساب (بدون كارت ائتمان) هي Geoapify.

التثبيت والتشغيل

ت المكتبات

```
npm install
```

يرات البيئة

انسخ `.env.example` إلى `.env` لأ القيم:

```
cp .env.example .env
```

المتغيرات المطلوبة:

```
PORT=3000
MONGODB_URI=<رابط اتصال MongoDB محلي أو - بتاعك Atlas>
GEOAPIFY_API_KEY=<مفتاح مجاني من geoapify.com>
```

> للحصول على `GEOAPIFY_API_KEY` : سجّل حساب مجاني على geoapify.com، اعمل Project جديد، وخد المفتاح من قسم "API Keys".

فيل السيرفر

```
npm run dev
```

يعمل السيرفر افتراضيًا على `http://localhost:3000` .

شكل ملف الإكسل المتوقع

المنتج	مكان
منتج 1	شارع طلعت حرب، وسط البلد، القاهرة
منتج 2	شارع المجاز، مصر الجديدة، القاهرة

- عمود الأماكن يُكتشف تلقائيًا (يقبل: مكان، المكان، عنوان، العنوان، الموقع، address، location).
- عمود المنتج (أو أي عمود آخر) يُتجاهل تمامًا.

ب أن تكون صفوفًا حقيقية (أسماء شوارع/مناطق مصرية)، وليست إحداثيات.

ي (المستخدم النهائي)

```
POST /api/process
```

Body (multipart/form-data):

المفتاح	النوع	الوصف
`file`	File	ملف الإكسل
`startLat`	Text	خط عرض نقطة بداية المندوب
`startLng`	Text	خط طول نقطة بداية المندوب

الرد (JSON):

```
{
  "message": "تمت معالجة الرحلة بنجاح...",
  "tripId": "...",
  "startPoint": { "lat": 30.0444, "lng": 31.2357 },
  "totalDistanceKm": 83.8,
  "orderedLocations": [ { "rawAddress": "...", "visitOrder": 0, "distanceFromPrevious": 6722 }, ... ],
  "failedLocations": [ { "rawAddress": "...", "reason": "..." } ],
  "pdfUrl": "/pdfs/trip-xxxx.pdf"
}
```

داخليًا: 1. قراءة الإكسل واستخراج الأماكن 2. تحويل العناوين إلى إحداثيات (Geocoding) — مع كاش لتفادي التكرار 3. ترتيب أفضل مسار زيارة (خوارزمية التحسين) 4. توليد PDF بالنتيجة الكاملة

> رابط الـ PDF نسبي — أضف عنوان السيرفر قبله عند الاستخدام: `http://localhost:3000` + `pdfUrl`.

نهاية إضافية (للاختبار والصيانة فقط)

مخصصة للاستخدام النهائي من المستخدم، لكنها مفيدة لتتبّع كل خطوة على حدة أو حل مشكلة بند واحد بدون إعادة الرحلة بالكامل:

Endpoint	الوظيفة
`POST /api/trips/upload`	رفع الإكسل فقط (بدون geocoding أو تحسين)
`POST /api/trips/:id/geocode`	تحويل عناوين رحلة موجودة لإحداثيات
`POST /api/trips/:id/optimize`	ترتيب مسار رحلة تم عمل geocoding لها
`GET /health`	التأكد أن السيرفر يعمل

هيكل المشروع

```
route-optimizer/
├── config/
|   ├── db.js              # الاتصال بـ MongoDB
|   └── multerConfig.js    # إعدادات رفع الملفات
├── controllers/
|   ├── tripController.js      # منطق الخطوات المنفصلة (upload/geocode/optimize)
|   └── processController.js   # منطق الـ endpoint الموحّد /api/process
├── models/
|   ├── Trip.js            # (سكيما الرحلة (الأماكن، المسار، الحالة
|   └── LocationCache.js   # كاش نتائج الـ Geocoding
├── routes/
|   ├── tripRoutes.js
|   └── processRoutes.js
├── utils/
|   ├── geocodeService.js    # تحويل عنوان لإحداثيات (Nominatim + كاش)
|   ├── routeOptimizer.js    # خوارزمية Nearest Neighbor + 2-opt
|   ├── staticMapService.js  # توليد رابط خريطة ثابتة (Geoapify)
|   ├── arabicTextService.js # تشكيل النص العربي لعرضه صح في PDF
|   └── pdfGenerator.js      # بناء ملف الـ PDF النهائي
├── fonts/
|   └── Amiri-Regular.ttf   # خط عربي مضمّن للـ PDF
├── public/pdfs/            # ملفات الـ PDF المولّدة (قابلة للتحميل عبر /pdfs/...)
├── uploads/                # (ملفات مؤقتة أثناء المعالجة (تُحذف تلقائيًا بعد القراءة
└── server.js
```

كيف تعمل خوارزمية ترتيب المسار

1. حساب المسافة الجغرافية الفعلية بين أي نقطتين. Haversine formula

2. Nearest Neighbor يبدأ من الموقع المندوب، وفي كل خطوة يختار أقرب نقطة لم تتم زيارتها بعد.
تحسين 2-opt يعكس أجزاء من المسار الناتج ليرى إن كانت المسافة الكلية ستقل، ويكرر ذلك حتى لا يوجد تحسين ممكن.

الحقيقي (TSP)، وهو سريع جدًا (أقل من ثانية) لعدد نقاط يصل حتى 200-300، وهو مناسب تمامًا لحجم الاستخدام الحالي (50-150 نقطة).

> المسار المفتوح (Open Path) أي من نقطة البداية ولا يعود إليها في النهاية.

بي في PDF

مكتبة `pdfkit` مصممة أصلًا للغات LTR ولا تحتوي محرك bidi حقيقي. تم حل هذا عبر: - تشكيل الحروف العربية (`arabic-reshaper`) حتى تظهر متصلة بشكل صحيح. - عكس ترتيب الكلمات يدويًا لعرض RTL صحيح.

النتيجة مقبولة جدًا لتقرير داخلي، لكنها ليست مطابقة 100% لجودة عرض المتصفح (قد تظهر مسافات ضيقة بين بعض الكلمات في العناوين الطويلة). إذا كانت مطلوبة جودة احترافية كاملة مستقبلًا، الحل الأدق هو توليد PDF عبر متصفح حقيقي (`puppeteer`) بدلًا من `pdfkit`

ملخص المفاتيح المطلوبة

الخدمة	مطلوب مفتاح؟	التكلفة
Nominatim (Geocoding)	لا	مجاني بالكامل
Geoapify (صورة الخريطة)	نعم (تسجيل مجاني)	3000 طلب/يوم، مجاني حتى 3000 طلب
MongoDB	لا (لو محلي) / connection string (لو Atlas)	مجاني

الخطوات القادمة (لم تُبنَ بعد)

- رفع الملف + عرض الخريطة التفاعلية (يُبنى بشكل منفصل من طرف الفريق).
- دعم مندوبين متعددين وتوزيع النقاط على أكثر من مسار في نفس اليوم.
- إعادة محاولة ذكية للعناوين التي تفشل في الـ Geocoding (تصحيح يدوي من الواجهة).