אפיון טכני: מערכת אוטומציות וזרימת נתונים – TopScan

סקירה כללית

מטרה

מסמך זה מתאר את עיצוב התוכנה של מערכת ניהול ההזמנות וזרימת הנתונים החדשה של החברה. המערכת מאגדת נתוני מכירות משני ערוצים (Amazon, Shopify) לבסיס נתונים מרכזי ב-AirTable, כאשר Make משמש כמנוע מרכזי להפעלה וניהול כל האוטומציות ו-Apify מטפל בסקרייפינג במקרים שבהם אין API זמין.

היקף

- שליפה אוטומטית יומית של נתוני Amazon מ-Apify/Puppeteer) ScaleInsights)
- שליפה של הזמנות ששולמו מ-Shopify דרך API
- אחסון מרכזי ב-AirTable עם לוגיקת מניעת כפילויות
- התראות WhatsApp דרך Twilio לדוחות

יעדים עסקיים

- **מקור נתונים יחיד ואמין** – כל ההזמנות מכל הערוצים במקום אחד
- **ללא התערבות ידנית** – סנכרון יומי אוטומטי מלא
- **נתונים מוכנים ל-BI** – מובנים ונקיים לדשבורדים ודוחות
- **תשתית מדרגית** – ניתן להרחיב בקלות ולהתפתח לכל דבר נוסף שתהיה אליו דרישה בעתיד

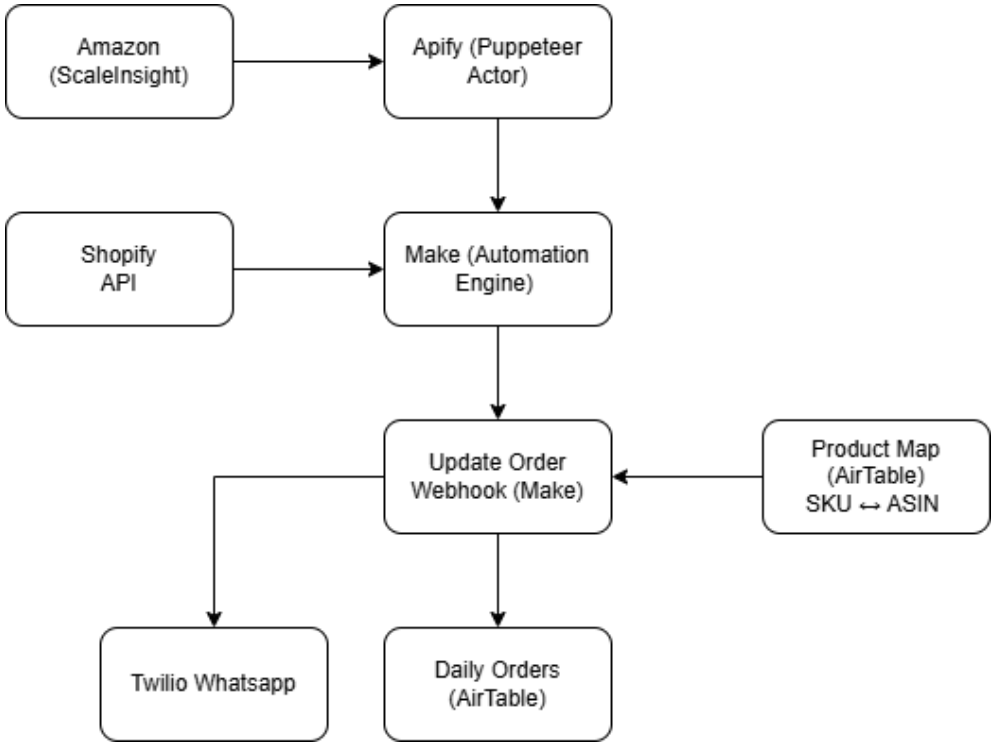
Tech Stack

Service	Role	Comments
Make	מנוע ראשי לניהול אוטומציות	המוח המרכזי של המערכת
AirTable	בסיס הנתונים המרכזי לכל ההזמנות	ניתן להעברה ל-PostgreSQL בהמשך
Apify	הרצת Puppeteer actors לסקרייפינג	שימוש מינימלי – רק כשאין API
ScaleInsights	מקור נתוני הזמנות Amazon	ייצוא קבצים יומי דרך Puppeteer

Shopify API	מקור הזמנות Shopify ששולמו	REST / GraphQL API
Twilio WhatsApp API	שליחת התראות וסיכומים יומיים	מומלץ כנקודת התחלה – קל לאינטגרציה

עיקרון עיצוב: עדיפות לאינטגרציות API ישירות על פני סקרייפינג. Apify/Puppeteer הוא אפשרות גיבוי בלבד.

ארכיטקטורה בסיסית של המערכת



עיצוב בסיס הנתונים

AirTable ייאפשר לגדול בצורה מובנית מהבסיס ועד לפתרון מלא ומתפתח, מבלי לנטוש את הכלי.

יכולת	תיאור
Interfaces	בניית דשבורדים ויזואליים ישירות על גבי הנתונים, ללא כלי BI חיצוני
Views	סינון, קיבוץ ומיון הנתונים לפי צרכים שונים (לפי מוצר, תאריך, שוק)
Automations	טריגרים פנימיים – לדוגמה: שליחת התראה כשנוספת רשומה מסוימת חדשה
Forms	יצירת טפסי קלט להזנה ידנית של נתונים במקרה הצורך
API מובנה	כל טבלה חשופה אוטומטית דרך REST API – קל לחיבור מכל מקום
Permissions	שליטה מדויקת מי רואה ומי יכול לערוך – לפי משתמש או תפקיד
Extensions	תוספים מובנים: גרפים, מפות, סיכומים סטטיסטיים ועוד

המלצה: בשלבים הראשונים, ניתן להשתמש ב-AirTable Interfaces כתחליף לכלי BI חיצוני. זה מפשט את הארכיטקטורה ומאפשר להתחיל לעבוד מהר.

טבלה: Product Map

ממפה מזהי מוצרים בין הפלטפורמות השונות (Amazon ו-Shopify). נדרשת מכיוון ש-Shopify אינו חושף ASINs – רק SKUs.

שדה	סוג	תיאור
ASIN	Text	מזהה מוצר ב-Amazon
SKU (short code)	Text	מזהה מוצר ב-Shopify
Product Name	Text	שם המוצר הגנרי

טבלה: Daily Orders

מאחסנת את כל ההזמנות היומיות מכל הערוצים.

שדה	סוג	תיאור
ASIN	Text	מזהה המוצר ראשי
Source	Text	מקור ההזמנה (/ amazon shopify)
Date	Date	תאריך ההזמנה
Marketplace	Single Select	שוק / ארץ (US/IT/...)
Units Sold	Number	כמות יחידות שנמכרו
Revenue	Currency	סכום הכנסה
Created At	DateTime	חותמת זמן יצירת הרשומה
...	...	...

מפתח ייחודיות

רשומות מזוהות באופן ייחודי לפי השילוב: **ASIN + Source + Date + Marketplace**

מפתח מורכב זה משמש בתור "המזהה" של כל הזמנה בתאריך מסוים ובאזור מסוים.

מפרט תהליכים

Webhook: Update Order

תהליך הכתיבה המרכזי. כל שאר ה-Scenarios מזינים נתונים לתוכו. אף תהליך אחר אינו כותב ישירות ל-Daily Orders.

טריגר: HTTP POST webhook (נקרא על ידי Scenarios אחרים ב-Make)

פורמט קלט:

```
{
```

```
  "asin": "XXXXXXX",
  "source": "amazon",
  "date": "2025-03-01",
  "marketplace": "US",
  "...":" ...",
  ...
}
```

לוגיקה:

1. קבלת ה-payload
2. חיפוש ב-Daily Orders לפי: ASIN + Source + Date + Marketplace
3. אם לא נמצא -> צור רשומה חדשה
   אם נמצא והנתונים זהים -> דלג
   אם נמצא והנתונים שונים -> עדכן את הרשומה הקיימת

Scenario: ScaleInsight Scraper

סנכרון יומי אוטומטי של נתוני Amazon מ-ScaleInsights דרך Puppeteer.

טריגר: מתוזמן – כל יום בשעה 10:01

שלבים:

1. Make – **Scheduled Trigger** מפעיל את ה-Scenario בשעה 10:01 מדי יום
2. **הרצת Apify Actor** – מפעיל את ה-Puppeteer Actor המוגדר מראש, מוריד את כל קבצי הדוחות ומאחד אותם ל-JSON אחיד
3. **Webhook סיום Actor** – בסיום, Apify שולח webhook חזרה ל-Make עם ה-JSON המאוחד
4. **פיצול ל-**Make – **Update Order** מבצע איטרציה על כל פריט ב-JSON ומעביר אותו ל-Webhook של Update Order

קבצי קוד רלוונטיים קודמים:

- index.js
- read-scaleinsight.js

Scenario: Shopify Watcher

שולף הזמנות ששולמו מ-Shopify וממפה אותן ל-ASINs לאחסון אחיד.

טריגר: מתוזמן יומית (10:01).

שלבים:

1. **שליפת הזמנות ששולמו** – משיכת כל ההזמנות ששולמו מ-Shopify לתקופה של שבעה ימים דרך API
2. **ריכוז לפי מוצר** – קיבוץ לפי SKU + Source + Date + Marketplace
3. **חיפוש ב-Product Map** - שופיפיי אינו חושף ASINs. לכל SKU, מחפשים את ה-ASIN המתאים בטבלת Product Map
4. **העברה ל-Update Order** – שליחת הרשומות המועשרות (כולל ASIN) ל-Webhook של Update Order

הערה: חלון ה-lookback של 7 ימים הוא רשת ביטחון לתפיסת הזמנות שהתפספסו או עודכנו. לוגיקת Update Order מטפלת בכפילויות – שליחה חוזרת של רשומות קיימות ללא שינוי היא בטוחה לחלוטין.

לוגיקת עדכון

ה-Webhook של Update Order *מממש* את עץ ההחלטות הבא עבור כל רשומה נכנסת:

תנאי	פעולה	סיבה
רשומה לא נמצאה	INSERT	נתון חדש – יצירת רשומה
רשומה נמצאה, נתונים זהים	SKIP	אין שינוי – חיסכון בקריאות API
רשומה נמצאה, נתונים שונים	UPDATE	נתון עדכני יותר – דריסה

מנגנון זה מבטיח כי הרצת אותו סנכרון פעמיים תייצר תוצאה זהה, ללא רשומות כפולות.

התראות – אינטגרציית WhatsApp

המערכת תשלח הודעות WhatsApp אוטומטיות דרך **Twilio** לבעלי עניין רלוונטיים.

יתרונות Twilio

- Sandbox חינמי לבדיקות
- מודול Native ב-Make (ללא HTTP מותאם אישית)
- תמיכה ב-Template Messages מאושרות Meta
- תמיכה ב-Webhook לקבלת תשובות
- קל ומהיר לקבלת אישור לשימוש ב-Make לעומת Meta API.

סיכונים ופתרונות

סיכון	חומרה	פתרון
מגבלת API של AirTable (5 req/sec)	בינוני	Rate limiting ב-Make + תור בקשות
שינוי UI ב-ScaleInsights שובר את Puppeteer	גבוה	שינוי של הUI עלול להקריס את הפעולה
מגבלות אחסון / שורות ב-AirTable	בינוני	תכנון מעבר ל-PostgreSQL Supabase בהמשך, או לAzure או לכל דבר רלוונטי אחר. בנוסף, אם באמת יוחלט להשתמש בדאשבורדים של Airtable, תוחלטנה חלוקה של מה נשמר בAirtable ומה במסד הנתונים הגדול.
אין מנגנון Retry בכשל	גבוה	הוספת מודולי Error Handler ב-Make

אבני דרך

שלב	תוצר	עדיפות
שלב 1 – בסיס	Update Order + ScaleInsights Scraper + Shopify Watcher	גבוה
שלב 2 – התראות ודו"ח יומי	אינטגרציית WhatsApp (סיכום יומי)	גבוה
שלב 3 – BI	דשבורדים ב-AirTable Interfaces; חיבור לכלי BI חיצוני (/ Looker Studio Power BI)	בינוני

שלב 4 – סקייל	מעבר ל-/ PostgreSQL Supabase אם AirTable אינו מספיק	נמוך

שאלות פתוחות

- האם AirTable מספיק לטווח הארוך, או שכדאי לתכנן מעבר ל-DB כבר עכשיו?
- מה תדירות הסנכרון הרצויה ב-Shopify?
- האם נדרש backfill היסטורי/טעינת הזמנות עבר ל-Daily Orders? כמה היסטוריה? היכן קיים המידע הנ"ל?