

פרויקט קורס: שרת מטרות צבאיות (Targets Server)



תיאור כללי

בפרויקט זה我们将构建一个名为 Express 的系统，用于管理军事目标。该系统将提供 CRUD 操作（创建、读取、更新、删除）。

- REST 和 HTTP
- Headers, Path Params, Query Params, Body
- Middleware
- 工作与 JSON
- Node.js File IO
- 稳定性、故障转移和负载均衡

⚠️ 项目的关键点在于：
在所有端点上：

- 所有参数均为字符串，并且必须为 JSON 格式
- 不使用 DB
- 不使用内存存储
- 所有端点都使用 Node.js fs 模块实现

מבנה נתונים — Target (מטרה צבאית)



每个目标表示为一个 JSON 对象。

```
{  
    "id": "t-102",  
    "codeName": "Iron Falcon",  
    "region": "north",  
    "priority": 4,  
    "status": "new",  
    "createdAt": "2025-12-21T09:30:00.000Z"  
}
```

פירוט שדות

תיאור	שדה
מזהה ייחודי של המטרה	id
שם קוד של המטרה	codeName
אזור פעילות: north, south, center, unknown	region
רמת חשיבות 1–5 (5 = הגבואה ביותר)	priority
new, approved, in_progress, completed	status
זמן ייצרת המטרה (ISO)	createdAt

מבנה קבצים מחיבר

```
/data
targets.json — L
{
  []
  : "targets"
}
```

פaza 1 — HTTP בסיסי וקריאה מקבצים (READ ONLY)

מטרת הפעלה

להבין:

- איך שרת מקבל מידע מהלקוח
- באילו דרכים שונות מעבירים מידע בבקשת HTTP
- למה `Header ≠ Path ≠ Query`
- איך לקרוא נתונים מקובץ JSON ולהחזיר תשובה

בפaza זו: 

- משתמשים רק ב-GET
- לא משנים קבצים

- כל נתון נקרא מ-`targets.json`.
-

תרגיל 1 — GET /health

בדיקה תקינות שרת

למה זה קיים?
זה endpoint בסיסי לניטור שירותי.
אפשר לבדוק שהשרתamazon ופועל.

תגובה

```
}
```

,"status": "ok"
"serverTime": "ISO_TIMESTAMP"

```
{
```

תרגיל 2 — GET /briefing

זיהוי הלוקוט באמצעות Header

למה **Header**?
זה מודיע שמצויה את השולח, לא את המשאבות.

Header חובה

Client-Unit: Golani

אם חסר → 400

תגובה

```
}
```

,"unit": "Golani"
"message": "briefing delivered"

```
{
```

תרגיל 3 — GET /targets/:id

שליפת מטרה לפי מזהה

למה **?Path Param**
כי ה-ID הוא חלק מהמצביע עליו.

מה השרת עושה

- קורא את `targets.json`
 - מחפש מטרה עם ID תואם
 - מחזיר או 404
-

תרגיל 4 — GET /targets —

שליפת אוסף מטרות + סינון

Query Params

- region
- status
- minPriority

למה **?Query**
כי אלו פילטרים על אוסף, לא מצביע אחד.

1 + אקסטרא פaza

GET /targets/:id/brief

תצוגה מצומצמת של מטרה (לא כל הנתונים)

GET /intel/ping

Header לזיהוי היחידה + Query לרמת פירוט

GET /targets/search

חיפוש חופשי לפי `q` (לא התאמת מדויקת)

פaza 2 — Middleware, JSON Body — וכתיבה לקבצים

מטרת הפaza

להבין:

- sh-endpoint לא אמר להכיל הכל
 - למה יש Middleware
 - איך שרת מקבל נתונים מורכבים
 - איך שינויים נשמרים לקובץ
-

תרגיל 1 — Middleware לוג

למה?

כדי להימנע מכפילויות ולרכז לוגיקה חוזרת.

מדים:

- method
 - path
 - timestamp
-

תרגיל 2 — Header middleware — שטוף

טוף לכל Response:

X-Server-Start-Time

למה ?Middleware
זה מידע תשתיתי, לא עסקי.

תרגיל 3 — POST /targets —

יצירת מטרה חדשה

למה ?
Body
כי יוצרים משאב עם כמה שדות.

Flow

1. בדיקת Content-Type
2. קריאת Body
3. ולידציה
4. קריאה מהקובץ
5. הוספה
6. כתיבה לקובץ

תרגיל 4 — PUT /targets/:id

עדכן מטרה

שילוב:

- Path Param (איזו מטרה)
- Body (מה משתנה)

+ אקסטרה פaza 2

POST /targets/:id/status

עדכן נקודתי של סטטוס

PUT /targets/:id/priority

עדכן עדיפות בלבד

POST /debug/echo

החזרת כל נתונים הבקשה לצורכי דיבוג

פaza 3 — CRUD מלא, יציבות ודיפלויימנט

מטרת הפaza

לבנות שרת:

- יציב
 - מתועד
 - עובד אחרי restart
 - מוכן לפרודקשן
-

תרגיל 1 — `DELETE /targets/:id`

מבחן מטרה מהקובץ

תרגיל 2 — Error Handling — מרכזי

למה קריטי?

- קריאה/ כתיבה לקובץ עלולה להיכשל
 - אסור לקלוט או לחשוף שגיאות פנימיות
-

תרגיל 3 — README

תיעוד:

- מבנה המטרה
 - כל ה-`endpoints`
 - דוגמאות שימוש
-

תרגיל 4 — Deployment

דרישות:

- PORT מתוך `env`
 - ייצור קובץ אם חסר
 - שרת עובד אחרי restart
-

+

אקסטרא פאג'ז 3

DELETE /targets

מחיקה לפי פילטרים בלבד (הגנה מפרודקشن)

GET /intel/summary

סיכום סטטיסטי במקומ raw data

POST /targets/bulk

יצירה מרובה + טיפול בשגיאות חלקיות

תוצר סופי

שרות Express.js מלא:

- CRUD
- File IO
- Middleware
- תיעוד
- רץ בפרודקشن

אם תרצה, השלב הבא יכול להיות:

- Rubric בדיקה מפורט לפיה פaza
- Checklist בדיקות לתלמיד
- גרסת מבחן עם דרישות מינימום
- Skeleton קוד התחלתי

פשוט וגייד.