

דו"ח הנדסה מחדש – המרת מערכת PHP לייעודית ל־WordPress

אתר: israeli-expert.co.il

כותרת רשמית: דו"ח המרה והטבות עסקיות־טכנולוגיות

הוכן עבור: עודד (בעלים/לקוח)

הוכן על ידי: pablo rotem

גרסה: Board / Director Level (ממוקדת יתרונות, עומק טכני בשפה פשוטה)

תוכן עניינים

1. תקציר מנהלים – מה עשינו ולמה זה חשוב

המערכת הוותיקה של האתר הייתה אפליקציית PHP ייעודית שנבנתה לאורך שנים כקוד 'חד-גושי' (Monolith) המשלב קבצי תצוגה (HTML) יחד עם לוגיקה עסקית וחיבורי מסד נתונים באותם קבצים. המרה ל-WordsPress אינה 'העתקת קבצים' אלא הנדסה מחדש (Re-Engineering): בנייה מחדש של תשתית התוכן, שכבת התצוגה, ניהול משתמשים והרשאות, אבטחה, וקווי הרחבה עתידיים – תוך שמירה על ההתנהגות העסקית של האתר.

התוצאה היא פלטפורמה מודרנית, סטנדרטית ומודולרית: קל יותר לתחזק, לגבות, לשחזר, להרחיב ולהעביר בין שרתים, עם קהילה עולמית של מפתחים ותוספים מוכנים שמקצרים זמן ועלות לפיצ'רים עתידיים.

1.1 תועלות מרכזיות (במילים פשוטות)

- יציבות ועדכונים: ליבת WordPress מתעדכנת באופן קבוע ומטפלת בפרצות אבטחה ידועות.
- אבטחה חזקה יותר: הרשאות מובנות, הגנות על טפסים, הקשחת קלט/פלט ומנגנוני מניעת מתקפות.
- ניהול תוכן קל: מערכת CMS שמאפשרת לצוות לא-טכני לערוך עמודים, טפסים ותכנים בלי לגעת בקוד.
- זמינות ומעבר שרת: גיבויים ושחזורים פשוטים יותר, ומעבר סביבה/שרת הופך פרויקט של שעות ולא ימים.
- פיתוח עתידי מהיר: פיצ'רים רבים מגיעים כתוספים (SEO, טפסים, חנויות, קאש, אבטחה) במקום לבנות מאפס.
- הפחתת תלות במפתח יחיד: קל למצוא מפתח WordPress בעתיד בשל היצע עצום של אנשי מקצוע.

2. תמונת מצב – לפני ואחרי ההמרה

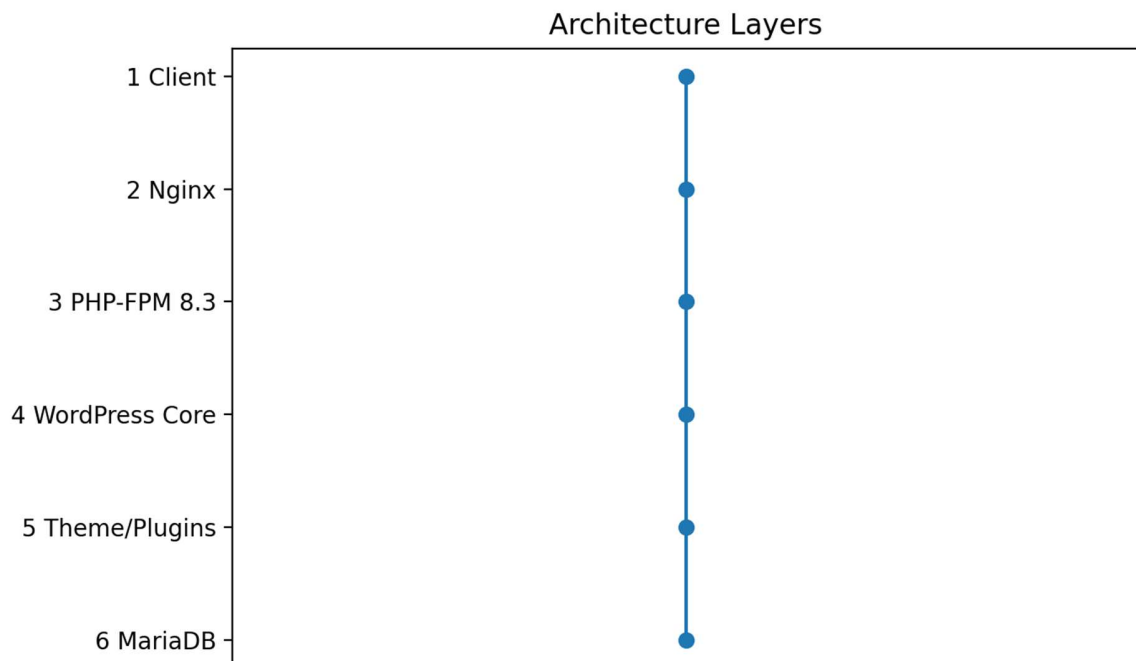
2.1 לפני: מערכת Legacy PHP

במערכת הישנה, כל מסך באתר היה לרוב קובץ PHP שמכיל ביחד: HTML (תצוגה), PHP (לוגיקה), ולעיתים שאילתות SQL (נתונים). מבנה כזה מקשה על תחזוקה ומעלה סיכון – כי שינוי קטן במסך אחד יכול להשפיע על אחרים, והאבטחה תלויה מאוד במשמעת הקוד של המפתח לאורך השנים.

2.2 אחרי: WordPress על PHP 8.3

לאחר ההמרה, האתר עובד כ'פלטפורמה': WordPress מספק ליבה יציבה (Framework), מערכת הרשאות, ניהול משתמשים, שכבת תבניות (Theme) להצגת האתר, ושכבת תוספים (Plugins) ללוגיקה עסקית. הקוד העסקי מופרד בצורה נקייה יותר, וניתן לשדרג רכיבים בלי לשבור את האתר כולו.

2.3 תרשים שכבות ארכיטקטורה



מקרא: (1) דפדפן משתמש, (2) שרת PHP-FPM, (3) Nginx שמריץ (4) PHP 8.3 ליבת WordPress, (5) תבנית ותוספים מותאמים, (6) מסד נתונים MariaDB/MySQL.

3. מה השתנה בפועל – פירוט תהליכי ההמרה

3.1 מיפוי מסכים ונתיבי מערכת (Routing)

במערכת PHP ישנה, הניווט מתבסס לרוב על קבצים ופרמטרים (לדוגמה `...=index2.php?id`). בוורדפרס, עובדים עם 'כתובות נקיות' (Permalinks) שמועילות ל-SEO ולחווית משתמש. במסגרת ההמרה בוצע מיפוי של כל המסכים והדפים, ונבנו תבניות עמוד (Page Templates) כך שהאתר מציג את אותם תכנים בכתובות מודרניות ועקביות.

3.2 שכבת תבנית (Theme) – בנייה מחדש של UI

תבנית בוורדפרס היא שכבת התצוגה: HTML/CSS/JS + קבצי PHP מינימליים שמושכים נתונים מהמערכת. במקום 'לזרוק' HTML בתוך קבצי לוגיקה, בונים רכיבים: header/footer, תבניות עמודים, תפריטים, וקומפוננטות. במסגרת העבודה הועבר העיצוב הישן למבנה תבנית תקני, כולל התאמות רספונסיביות (מובייל), RTL בעברית, וטיפול בתפריטים (המבורגר) והתנהגויות שונות בדסקטופ/מובייל.

3.3 שכבת תוספים (Plugins) – לוגיקה עסקית

כאשר יש פונקציונליות שהיא 'מערכתית' (למשל טפסים, הרשמה, ניהול נתונים, חיבורים חיצוניים), נכון למקם אותה בתוסף ולא בתבנית. כך אפשר להחליף עיצוב בלי לאבד לוגיקה. בהמרה הופרדו חלקים עסקיים מהתצוגה והועברו לתוספים (או פונקציות מודולריות), כדי לשמור על תחזוקה נכונה ויציבה.

3.4 מסד נתונים – עבודה עם מבנה ישן בתוך פלטפורמה חדשה

המערכת כללה מסד נתונים קיים. בוורדפרס קיימות טבלאות ברירת מחדל (posts, postmeta וכו'), אבל כאשר יש טבלאות עסקיות קיימות – ניתן להשאיר אותן ולהתחבר אליהן בצורה בטוחה ומבוקרת. בוצע חיבור מחדש למסד הנתונים, תוך טיפול בקידוד (UTF-8), אינדקסים, ושכבת גישה שמונעת שאילתות לא בטוחות.

4. אבטחה – מה הוספנו ולמה זה קריטי

ההבדל הגדול בין מערכת PHP ייעודית ל-WordPress הוא סטנדרטיזציה של אבטחה. בוורדפרס קיימים מנגנונים מובנים להרשאות, אימות פעולות, והפרדת תפקידים – אך עדיין חשוב ליישם נכון בקוד מותאם.

4.1 Nonce – מה זה ולמה השתמשנו בזה

Nonce הוא 'אסימון חד-פעמי' (Token) שמצורף לטופס או פעולה רגישה. הוא מאפשר לוורדפרס לוודא שהבקשה יצאה מתוך האתר (ולא מאתר חיצוני שמנסה להפעיל פעולה מאחורי הקלעים). זה מגן מפני מתקפת CSRF – מצב שבו משתמש מחובר מבצע פעולה בלי שהתכוון.

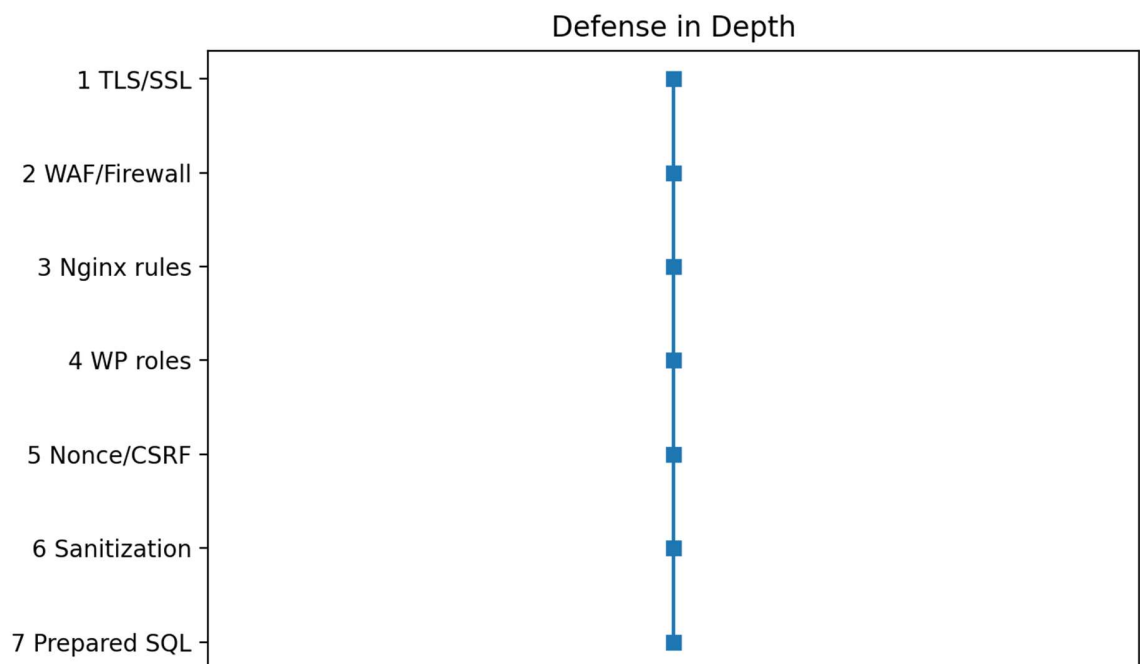
4.2 Sanitization ו-Escaping – ניקוי קלט וסינון פלט

Sanitization (ניקוי קלט) = כל מה שהמשתמש מזין (טקסט, מספר, אימייל) נבדק ומנוקה לפני שמירה למסד. Escaping (סינון פלט) = כאשר מציגים נתונים בדף, מסננים תווים כדי למנוע הזרקת קוד. שילוב של שני הדברים מגן מפני XSS – הזרקת JavaScript זדוני לדפים.

4.3 Prepared Statements – הגנה מפני SQL Injection

Prepared Statements הם דרך להריץ שאילתות בצורה בטוחה: במקום לחבר מחרוזות SQL ידנית, שולחים 'תבנית' ושמים ערכים כפרמטרים. זה מונע מצב שבו משתמש מכניס קלט שמזיז את השאילתה ומדליף נתונים.

4.4 תרשים שכבות הגנה (Defense in Depth)



מקרא: שכבות הגנה מצטברות – גם אם שכבה אחת נפגעת, אחרות ממשיכות להגן.

5. ביצועים, מהירות ו-SEO – יתרונות ממעבר לפלטפורמה מודרנית

5.1 PHP 8.3 – למה זה חשוב

PHP 8.3 מספק שיפורים במהירות הרצת קוד, ניהול זיכרון, ותמיכה ארוכת טווח. במערכות ישנות, לעתים נשאים על גרסאות PHP מיושנות בגלל תאימות – וזה סיכון אבטחתי. המעבר מאפשר לעבוד על גרסה עדכנית, לקבל תיקוני אבטחה, ולהריץ קוד מודרני בצורה יציבה.

5.2 קאש (Cache) – איך זה מקצר זמני טעינה

קאש הוא מנגנון שמאפשר לשמור תוצאה של דף/שאלתה כדי לא לחשב מחדש בכל בקשה. בוורדפרס ניתן להשתמש בקאש ברמת שרת (למשל Varnish/Nginx) וברמת אפליקציה (Plugins). התוצאה היא מהירות גבוהה יותר, במיוחד בעומסים ובמובייל.

5.3 SEO – תשתית טבעית טובה יותר

WordPress בנוי עם סטנדרטים שמקדמים SEO: כתובות נקיות, היררכיה של עמודים, תגיות, ומערכת תוספים עשירה שמאפשרת ניהול Titles/Meta, מפות אתר Schema, Sitemap) ועוד – ללא פיתוח מאפס.

6. תפעול, גיבויים, שחזורים והעברת שרת – למה זה נהיה קל יותר

במערכת ייעודית, גיבוי ושחזור תלויים בפרוצדורות ידניות ובידע ספציפי. בורדפרס יש שיטות עבודה סטנדרטיות: גיבוי קבצים + DB, סביבת Staging, וגם כלים שמכירים את המבנה ומקצרים תקלות.

6.1 מה זה Staging ולמה זה משנה

Staging היא סביבת 'הדמיה' – עותק של האתר שבו בודקים שינויים לפני שמעלים לייצור (Production). ככה מצמצמים סיכונים ומונעים נפילות אתר.

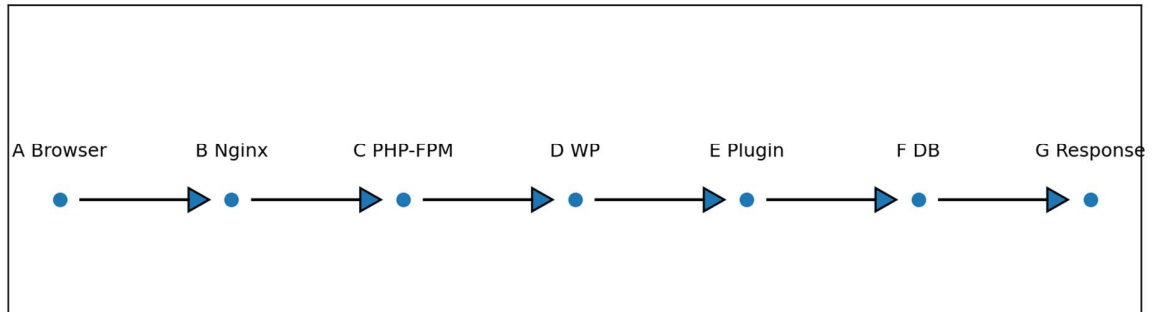
7. אפשרויות שדרוג עתידיות – למה עכשיו קל וזול יותר

כעת ניתן להוסיף יכולות בלי לבנות הכל מאפס. דוגמאות:

- WooCommerce – הוספת מוצרים/תשלומים בצורה תקנית.
- תוספי SEO – ניהול מתקדם של כותרות, תיאורים, Schema, מפות אתר.
- תוספי אבטחה – Firewall יישומי, ניטור קבצים, חסימת ניסיונות התחברות.
- תוספי טפסים – טפסי הרשמה מורכבים עם אימותים, אוטומציות ומיילים.
- Multilingual – הוספת שפות וניהול תרגום בצורה מסודרת.
- Analytics – חיבור מדידה, אירועים ומעקב המרות.
- Performance – Lazy-load, אופטימיזציית תמונות (WebP/AVIF), קאש מתקדם.

8. זרימת בקשה במערכת החדשה – מה קורה כשמשתמש נכנס לאתר

Request / Response Data Flow

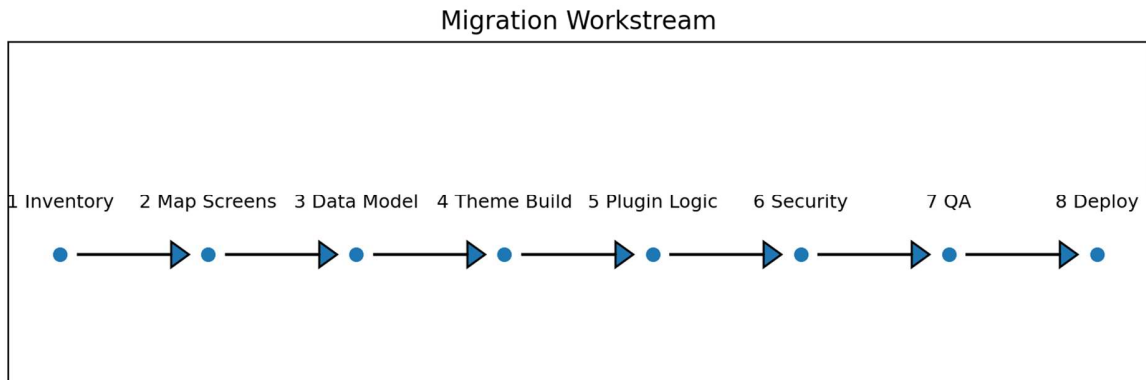


מקרא: A המשתמש מבקש דף, B Nginx מקבל, C PHP-FPM מריץ, D WordPress מנהל, E התוסף/התבנית בונים תשובה, F מסד נתונים מספק מידע, G נשלחת תגובה לדפדפן.

9. למה זה לקח זמן – פירוק מורכבות והקטנת סיכונים

המרה איכותית דורשת סדר פעולות: להבין את המערכת הישנה (כולל חריגים וטלאים), למפות פונקציות עסקיות, לייצר תבניות ותוספים תואמים, לבצע התאמות RTL/מובייל, להטמיע אבטחה מודרנית, ולבדוק התאמות מול תכנים קיימים. המטרה היא לא רק 'שיעבוד' – אלא שיהיה יציב, בטוח וניתן להמשך פיתוח לשנים קדימה.

9.1 תרשים תהליך עבודה (Workstream)



כל שלב הוריד סיכונים והבטיח שההתנהגות העסקית נשמרת תוך מעבר לתשתית מודרנית.

10. מילון מושגים – הסבר פשוט (הרחבה)

- CMS – מערכת ניהול תוכן שמאפשרת לערוך עמודים ותכנים בלי קוד.
- Theme – תבנית העיצוב וההצגה של האתר (איך האתר נראה).
- Plugin – תוסף שמוסיף יכולות לאתר (איך האתר מתנהג).
- Hook/Filter – נקודת חיבור שמאפשרת להוסיף לוגיקה בלי לשנות קוד ליבה.
- Nonce – אסימון חד-פעמי שמאמת פעולות ומונע בקשות זדוניות.
- CSRF – מתקפה שמנסה לגרום למשתמש לבצע פעולה בלי שהתכוון.
- XSS – הזרקת קוד זדוני לדף דרך קלט לא מנוקה.
- Sanitization – ניקוי/אימות קלט לפני שמירה.
- Escaping – סינון תווים בעת הצגת נתונים בדף.
- Prepared Statements – הרצת שאילתות בצורה בטוחה שמונעת SQL Injection.
- PHP-FPM – מנגנון שמנהל תהליכי PHP בצורה מהירה ויעילה.
- Permalinks – כתובות נקיות וקבועות שמועילות למשתמש ול-SEO.
- Staging/Production – סביבת בדיקות לעומת סביבת אתר חי.

11. המלצות והמשך הדרך (Roadmap)

המלצות לטווח קצר (0–30 יום):

- הקשחה נוספת של הרשאות וסקר תוספים פעילים.
- יצירת נוהל גיבוי ושחזור כולל בדיקת שחזור רבעונית.
- הפעלת קאש ברמת שרת/אפליקציה בהתאם לצורך.
- סגירת פערים אחרונים ב־SEO: Sitemap, Schema, Redirects.

המלצות לטווח בינוני (1–3 חודשים):

- הוספת דוחות אנליטיקה והמרות.
- שיפור מהירות מובייל: אופטימיזציה לתמונות, Lazy-load, מיניפיקציה.
- בחינת הוספת תוספי אבטחה וניטור.

המלצות לטווח ארוך (3–12 חודשים):

- הוספת מודולים חדשים באמצעות תוספים (למשל WooCommerce במידת צורך).
- בחינת Headless / API לפי צרכים עסקיים.
- בניית אוטומציות תפעול (ניטור, התראות, לוגים).