איך נראה פרויקט קסטום כשמסתכלים עליו דרך משקפי הספרינטים

פרויקט בניית אתר Custom שנכתב בקוד נקי ומתוכנן מראש מתנהג אחרת לגמרי ממערכת מדף. הוא לא נסמך על תבניות שמגבילות אותך, והוא לא כופה עליך פשרות בארכיטקטורה או בביצועים. מצד שני, הוא דורש ניהול קפדני, חלוקה חכמה לספרינטים, והבנה של התלויות בין UX, פיתוח, תוכן ו-SEO. מתוך פרויקטים שבנינו עבור עסקים שחיפשו אתר מותאם אישית, שמוכן לקידום ושאינו מבוסס וורדפרס, גיבשתי מפת דרכים פרקטית ספרינט אחר ספרינט. היא מכסה תכנון, פיתוח אתרים בקוד, בדיקות, השקה ואופטימיזציה, כולל היבטים של מהירות, אבטחה, מבנה קישורים פנימי וזרימת Link Juice.

המטרה היא לא רק לסיים אתר שעובד, אלא לבנות נכס דיגיטלי שניתן להרחיב, לתחזק, ולקדם לאורך שנים. אם אתם מתלבטים בין בניית אתר מאפס בקוד לבין מערכת מוכנה, או שואלים כמה עולה לבנות אתר בקוד, התהליך הבא ייתן פרספקטיבה תפעולית שמתרגמת החלטות ארכיטקטורה לערך עסקי.

פאזה מוקדמת: אפיון חד שמקטין סיכונים

השלב שקודם לספרינט הראשון קובע את הדי.אן.איי של הפרויקט. כאן מגדירים יעדי עסק, מיפוי עמודי ליבה, היררכיית תוכן ראשונית, וכללי SEO בסיסיים. אתר חכם מותאם לקידום לא נולד בסוף הפרויקט אלא ביום הראשון. אם יודעים מראש שנדרש אתר בקוד שמוכן ל-SEO, קובעים כללים לסרגל ניווט, עומק עמודים ושימוש נכון בכותרות H1-H3. ברמה הטכנית כדאי לקבל החלטות סביב סביבת פיתוח, רפוזיטורי Git, CI פשוט לבניית גרסאות, ובחירת סטאק - למשל React או Next.js עבור פרונט, Node או Python בשרת, ומסד נתונים שמתאים לדפוסי המידע.

כשלקוח מגיע עם דרישה של בניית אתר שלא בוורדפרס, אני בודק האם יש צורך בממשק ניהול פשוט או שמספיקה אינטגרציה לתשתית קיימת. לפעמים פתרון Headless CMS בקוד פתוח כמו Strapi נותן איזון יפה בין גמישות לבין מהירות פיתוח. היתרונות של בניית אתר בקוד פתוח הם שקיפות, קהילה רחבה ורכיבי אבטחה שנבדקים כל הזמן. מצד שני, אם יש דרישות מיוחדות לביצועים או הרשאות, פיתוח אתר בהתאמה אישית ללא שכבות מיותרות נותן שליטה מלאה.

ספרינט 1: UX, תוכן ואסטרטגיית SEO מתואמים

הספרינט הראשון מתכנס סביב שלושה זרמים: מפת אתר, אבטיפוסי UX קלילים, ועקרונות תוכן המתורגמים למבנה SEO. במקום מסמכים כבדים, אני מעדיף סקיצות אינטראקטיביות קצרות, שבהן בוחנים את הזרימה בין הדפים, את הסרגל הראשי ואת הדפוס של קישורים פנימיים. אתר Custom עם מבנה קישורים פנימיים חכם מתחיל במפות זעירות שרושמות מה יפנה למה, באילו עוגנים, ואיך ישתלבו רכיבי תוכן עשירים.

זה גם המקום להחליט על מסגרות טקסט: עומק פסקאות, שימוש ברכיבי Microcopy, איך נשלב טפסים או אזורי הרשמה. מבחינת מיתוג, מגדירים עקרונות טיפוגרפיה וצבעים אך לא ננעלים על פיקסלים. אם מדובר בבניית אתר עסקי עם React או Next.js, כדאי כבר עכשיו לקבע עקרונות של רספונסיביות ותמיכה ב-RTL, כולל בדיקה ידנית בשני מסכים ושני דפדפנים כדי לגלות מוקדם עיוותים.

ספרינט 2: בחירת סטאק והנחת תשתית נקייה

פיתוח אתר מתקדם בהתאמה מלאה דורש שכבת בסיס יציבה. בשלב זה אני מקים מונורפו או רפוזיטוריז נפרדים לפרונט ולבקנד, מגדיר Linting, פורמט אחיד בקוד, בדיקות יחידה מינימליות והמכונה הראשונה ב-CI. אתר בקוד נקי לא מגיע במקרה, הוא תוצר של שגרות קטנות שמשתלמות בהמשך. אם הולכים על Next.js, בוחרים בין SSR ל-SSG לפי סוג התוכן והצורך בעדכונים. אתר מהיר בקוד נקי לעיתים נשען על SSG עם ריענון אינקרמנטלי וכללי קאשינג חכמים.

חשוב לא להעמיס ספריות. בניית אתר חכם בקוד מתחילה בפלנקים פשוטים: רכיב תפריט, גריד, טיפוגרפיה, כפתורים, וטופס בסיסי. בשלב זה גם מגדירים תשתית SEO מובנה: תגיות מטא, סכמות JSON-LD מובנות לעמודי מוצר או מאמר, ויצירת Slugs הגיוניים. מי שעבד שנים עם וורדפרס מתרגל לפלאגינים שמוסיפים הכל בלחיצה, אך פה ההחלטה מודעת.

למה עדיף אתר בקוד על מערכת מוכנה? כי כתיבת התכונות הדרושות בלבד מונעת נפיחות, מאיצה את האתר ומקלה על תחזוקה.

ספרינט 3: קופסאות תוכן, ניווט והיררכיה פנימית

כאן אנו בונים את הארכיטקטורה של התוכן. אם הפרויקט כולל בלוג, אזור ידע, קטלוג שירותים או עמודי מקרה בוחן, המודלים מקבלים צורה. בפועל, זה אומר שמודל "מאמר" יכיל כותרת, תקציר, גוף, תמונה, קטגוריה, תגיות, עוגני H2, ותיבות קידום פנימיות שמציעות קריאה נוספת. פיתוח אתר חכם לקישורי SEO מתייחס כבר עכשיו לשאלה מי מקשר למי, באילו עוגנים, ובאיזה עומק כדי להעביר Link Juice אל עמודי כסף.

ללקוחות אני מסביר את ההבדל בין בניית אתרים בקוד לעומת פלטפורמות: המערכת לא תפתיע אותך עם מבנה URL מוזר או שינויי כותרות לא צפויים. אתה זה שקובע סטנדרטים, למשל /blog/topic-slug עם קטגוריות נקיות, או /services/vertical כדי למקד סמכות. בניית אתר בקוד שמוכן ל-SEO מאפשרת לנסח ידנית מפת אתר דינמית שתוצג לבוטים בדיוק כפי שנכון לעסק.

ספרינט 4: פיתוח רכיבי UI ובדיקות שימושיות

רכיבי UI נבנים כמודולים ניתנים לשימוש חוזר. לא כל רכיב מצדיק ספרייה, ולעיתים עדיף להישען על HTML, CSS ו-JS נקיים, במיוחד כשמדובר באזורים בעלי פרפורמנס קריטי. עבור מי שמבקש מדריך לבניית אתר HTML CSS JS, זה הרגע להתעקש על שכבות ברורות: סגנון מופרד מלוגיקה, נגישות כקו מנחה, ותמיכה מלאה במקלדת וקוראי מסך. אתר מותאם אישית לא יכול להרשות לעצמו להתעלם מהנחיות נגישות, בעיקר כי זה משפיע באופן ישיר על SEO ועל חוויית משתמש.

הבדיקות בספרינט זה הן בעיקר חווייתיות. אני מזמין שניים עד שלושה משתמשים מייצגים, נותן להם משימות קצרות, ומודד זמן מציאת מידע, טעינה ראשונית, והאם הטופס מרגיש אמין. השינויים שמגיעים מהתרגיל הזה בדך כלל יותר זולים בספרינטים מוקדמים, בעוד שבסוף הפרויקט הם כואבים ומאטים את ההשקה.

ספרינט 5: שכבת תוכן ועריכה עם כללי SEO פרקטיים

ברגע שהרכיבים יציבים, מתחילים להזין תוכן אמיתי. אם מדובר בבניית אתר לעסק בקוד, אני מדריך את הכותבים על מבנה פסקאות, עוגנים פנימיים, ותיאורים חלופיים לתמונות. אתר בקוד עם SEO מובנה יודע להציע שדות הכרחיים לעריכה, למשל Title Tag, תיאור מטא ידני, ומדד קצר ל-Readability שנותן כיוון בלי להפוך לכלי מכני. כאן גם מוודאים שכל עמוד מורכב סביב כוונות חיפוש ולא סביב מילים בודדות.

בהיבט טכני, מנגנוני תקצור תמונות ויצירת גרסאות WebP ודחיסה ללא אובדן נראים קטנים אך חוסכים עשרות אחוזים בזמן הטעינה. אם בחרנו ב-Next.js, ממנפים Image Optimization מובנה או שירות CDN לתמונות. בניית אתר מהיר בקוד נקי מתבססת על משמעת: לא מעלים וידאו במשקל 50MB לדף בית, לא טוענים ספרייה של 300KB בשביל אנימציה מינורית.

ספרינט 6: תשתיות אבטחה והקשחה

אבטחה היא החלטות קטנות רבות. בניית אתר מאובטח בקוד כוללת הגנה מפני XSS, CSRF, ולוגים סדורים לכל נקודת API. אם יש טפסים, מוסיפים Rate Limiting בצד השרת, ואם יש אזורי ניהול, אז 2FA ו-IP allowlist לפי הצורך. חשוב לקבוע מדיניות Content Security Policy מוקדם, גם אם בהתחלה היא גמישה ואז מתהדקת. חלק מהלקוחות מופתעים כשאני מציע לייצר חשבונות הרשאה ברמות שונות כבר לפני ההשקה. זה חוסך טעויות עריכה ומגן על שכבות קריטיות.

למי ששואל מה ההבדל בין וורדפרס לקוד בהקשר אבטחה, התשובה היא שליטה. בוורדפרס האבטחה תלויה בעשרות תוספים ובמשמעת עדכונים. כשבונים אתר בקוד אישי או בהתאמה מלאה, הדפנות ברורות, התלויות מצומצמות, והסיכון

לפרצות דרך רכיבים ישנים קטן. מצד שני, האחריות עליכם. אין עדכון אוטומטי שיציל לילה רע, ולכן תהליך ניהול גרסאות והקשחה הוא חלק מהשגרה.

ספרינט 7: אופטימיזציית מהירות, מדדים ובנצ'מרקים

כאן עוברים ממדיניות לביצוע. מודדים LCP, INP ו-CLS בכל סוג עמוד, מבודדים צווארי בקבוק, ומשפרים שלושה דברים קלאסיים: גודל חבילות JS, אסטרטגיית טעינת פונטים ותמונות, וקאשינג בצד השרת והדפדפן. פיתוח אתרים בקוד נותן לכם חופש לבחור האם לצרוב את הקריטיים ל-HTML כדי לקצר זמן לרנדר הראשון, או לדחות רכיבים לא חשובים לאחר אינטראקציה.

בפרויקט טיפוסי הצלחנו להוריד ג'אווהסקריפט נטען מ-420KB לכ-120KB פשוט על ידי פיצול חכם, החלפת ספריית תאריכים כבדה בפונקציות ייעודיות, והוצאת קוד צד שלישי שלא תרם ליעדים. זה לא קסם, זו מלאכת כירורגיה. אתר חכם מותאם לקידום מרוויח ממהירות לא רק בדירוגים, אלא גם בשיעור המרות שנשאר גבוה במובייל חלש.

ספרינט 8: אנליטיקה, אירועים ותשתית Data

לפני השקה רצינית, מיישרים קו על מדידה. לא כל אתר צריך 30 אירועים, אך כל אתר צריך למדוד CTA עיקריים ומיקרו אינטראקציות שמנבאות כוונה. מגדירים אירועים דרך Tag Manager או ישירות בקוד, תוך שמירה על פרטיות והסכמה. אם האתר פועל בשווקים שמחמירים במדיניות פרטיות, בונים מצבי Degraded שבו תכנים נטענים גם בלי Cookies מיותרים.

אני נוהג לבנות דשבורד אחד פשוט במערכת הניטור שמרכז נתוני SEO טכני, ביצועים והתנהגות משתמש. לא בשביל להיות Data-driven באקדמיה, אלא כדי לאתר במהירות איפה לגעת בספרינטים הבאים.

ספרינט 9: בדיקות אינטגרציה וקצה לקצה

אם יש חיבורי צד שלישי - CRM, סליקה, מערכות דיוור - זה הזמן לאשש תרחישים. בדיקות קצה לקצה עם נתוני בדיקה אמיתיים מציפות בעיות הרשאות, עיכובים ובעיות כתיבות API. חשוב להקפיד על ניהול סביבות: Dev, Staging ו-Production, עם נתיבי פריסה ידועים. בניית אתר בקוד לעסקים נמדדת ביכולת להתנהל בביטחון גם ביום עומס וגם ביום תקלה. ללא Staging ברור, הסיכון לעלות עם באג חמור פשוט גבוה.

ספרינט 10: תפעול השקה והקשבה ראשונית

ההשקה אינה אירוע חד פעמי. אני מעדיף שחרור מדורג: תחילה לקבוצה מצומצמת, ואז הרחבה. מגדירים Feature Flags שמאפשרים ביטול מהיר של חלקים בעייתיים, ומכינים חבילות רגרסיה קטנות. מבחינת SEO, מגישים Sitemaps, מוודאים שאין noindex דבוק בטעות, ומנטרים Index Coverage ביומיים הראשונים. אתר בקוד שמוכן ל-SEO זוכה להאצה מוקדמת אם מבנה הקישורים הפנימי כבר מקדם עמודי יעד ומונע יתמות של דפים.

ספרינט 11: תחזוקה מתמשכת ו-Roadmap קדימה

לאחר שהאתר באוויר, נכנסים לשגרה של שיפורים קטנים. שירותי פיתוח אתרים לא מסתיימים בהשקה, הם עוברים למוד של אופטימיזציה. בוחנים בקשות משתמשים, זמן טעינה לאחר הכנסת תכנים חדשים, והתנהגות של זחלנים. מייצרים Sprint Backlog למסלולי צמיחה: מיקרו-פיצ'רים חכמים, הרחבת דפי שירות, הוספת Sectionי מדריך שמביאים טראפיק אורגני. פיתוח אתר חכם לקישורי SEO לא נגמר בבניית קישורים חיצוניים, אלא בתיקון פנימי מתמשך שמכוון את זרימת Link Juice לפי יעד עסקי משתנה.

מתי לבחור קוד פתוח ומתי לבנות לגמרי מאפס

יתרונות בניית אתר בקוד פתוח ברורים: זמן פיתוח קצר יותר, קהילה, תוספים נקודתיים. במקרים שבהם צריך ממשק ניהול קל לתוכן שוטף, או חיבור מהיר ל-SSO, מערכת פתוחה נותנת מענה. מצד שני, כשביצועים, אבטחה מוקשחת, או חוויית משתמש ייחודית הם קריטיים, פיתוח אתר בהתאמה אישית, ללא שכבות כלליות, יעניק תוצאה נקייה ומהירה יותר. חברה לבניית אתרים בקוד צריכה להציג שקיפות על תלות בספריות ולתעד את ההשלכות ארוכות הטווח של כל בחירה.

יש גם היבט תפעולי: אם הארגון שלכם מחזיק צוות פנימי, נוח יותר לאמץ סטאק נפוץ עם קהילה פעילה. אם אין צוות, אבל יש צורך בביצועים חדים ובמינימום תחזוקה, ארכיטקטורה מצומצמת עם כמה שפחות תלויות תאפשר שקט תפעולי.

עלויות, תמחור ותיאום ציפיות

השאלה כמה עולה לבנות אתר בקוד לא נענית במספר אחד. פרויקט בסיסי לעסק קטן, עם 8 עד 12 עמודים, עיצוב מותאם, ו-SEO טכני תקין, עשוי לנוע בטווחי מחיר של עשרות אלפי שקלים, בהתאם למורכבות האינטגרציות והיקף התוכן. פרויקטים מורכבים עם קטלוגים, דשבורדים פנימיים או אזורי משתמשים נעים גבוה יותר. מחיר בניית אתר בקוד נקבע לפי סך הספרינטים, כמות האיש צוות, ומרווחי הבדיקות.

הצעת מחיר טובה תפרוט שלבי עבודה למסגרות זמן, תקצה מרווחי בדיקות, ותגדיר מה נחשב Change Request. מפתח אתרים בהתאמה אישית שמנהל נכון את הספרינטים יוביל אתכם להחלטות שמקטינות עלויות עתידיות, למשל פיצול פיצ'ר ל-MVP והשלמתו בספרינט שאחרי ההשקה.

השוואה שקולה לעומת וורדפרס ומערכות מוכנות

וורדפרס velolinx.co.il מצוינת לפרויקטים מסוימים. אם נדרש אתר תדמית מהיר עם תבנית איכותית, או בלוג פשוט, יש הגיון בבחירה. השאלה מה ההבדל בין וורדפרס לקוד הופכת מעניינת כשהאתר צריך לבצע מהר, לתמוך בזרימות משתמש ייחודיות, ולעבור ביקורת אבטחה קפדנית. בניית אתר שלא בוורדפרס מאפשרת הנדסה מוקפדת: אין שכבת תוספים שמביאה איתה קוד לא נחוץ, ואין תלות בעדכונים שעשויים לשבור רכיבים.

מבחינת SEO, אתר בקוד עם SEO מובנה מאפשר שליטה מלאה ב-Rendering, בהטמעת Schemas, ובקצבי עדכון Sitemap. מבחינת ביצועים, שליטה בחבילות ובפיצול קוד מעניקה יתרון משמעתי על פני פרויקטים שמעמיסים תוספים.

מפת התקדמות מבוססת ספרינטים - מה אתם מקבלים בפועל

ברמה היומיומית, חלוקה לספרינטים הופכת את הפרויקט לשקוף. כל שבועיים אתם רואים גרסה, מודדים, ומשנים כיוון אם צריך. אם יש תעדוף מחדש - לדוגמה, להקדים עמודי שירות שדרושים לקמפיין - מגלגלים אותם לספרינט הקרוב בלי לפרק את כל הפרויקט. כך בניית אתר מאפס בקוד נשארת נשלטת, גם כשמגיעים רעיונות חדשים באמצע הדרך.

מקרי קצה שחשוב להכיר

יש פרויקטים שבהם התוכן מתעכב. במקרה כזה אני מתכנן ספרינט טכני שמתקדם בלי לשבור את הלו"ז: שיפורי ביצועים, בדיקות רספונסיביות, או חיבורי אינטגרציה. יש גם מקרים שבהם מערכת החוץ משנה API. כדי שלא ניתקע, מוסיפים שכבת Adapter שמבודדת שינויי צד שלישי. ואם צוות העיצוב רוצה חופש אנימציות נרחב, מגדירים תקציב ביצועים קשיח: כמה מילישניות מותר לכל טרנזישן, ומה נדחה ל-Interaction.

דוגמאות מהשטח: החלטות קטנות שמשנות תמונה

באתר ידע למקצוענים, הקטנטנה שהכריעה הייתה עוגני H2 גלויים בתוכן עם קישורי "לקריאה קשורה" שמשנים כיוון לפי הקשר. התוצאה הייתה ירידה של 15 עד 25 אחוז בקפיצה לאחור ועלייה בזמן שהיה. בקטלוג שירותים, החלטה להפריד

ביקורות משתמשים ל-Microservice קטן איפשרה קצב טעינה יציב גם כשיש עשרות קריאות מקבילות. ובפרויקט עם שפות מרובות, הימנעות מקוד משותף מיותר בין RTL ל-LTR חסכה לנו שבועי תיקונים.

צ׳ק-ליסט קצר לפני שמתחילים

- יעדים עסקיים וסרגל מדדים ברור: המרות, לידים, או צריכת תוכן.
- בחירת סטאק בהתאם לצרכים, לא לפי אופנה. React או Next.js כשיש היגיון.
- החלטות SEO מוקדמות: היררכיה, Slugs, ומבנה קישורים פנימי.
- כללי אבטחה וקשיחות מינימליים מוגדרים מראש, כולל CSP ו-Rate Limiting.
- תוכנית ספרינטים עם נקודות בדיקה וגרסאות ביניים נראות לעין.

היגיון כלכלי: השקעה שמחזירה את עצמה

כשבודקים עלות מול תועלת, בניית אתר Custom לא נמדדת רק במחיר פתיחה. אתר נקי, מהיר ובטוח, שמביא טראפיק אורגני יציב, חוסך הוצאות בפרסום, מצמצם נטישה, ומייצר חוויית מותג. בניית אתר בקוד לעסקים שמבקש גדילה מדויקת, מאפשר הוספת פיצ'רים ממוקדים בלי להיתקע בתקרה של מערכת מוכנה. לאורך זמן זו אסטרטגיה שמאפשרת צמיחה לפי מפת דרכים, לא לפי אילוצי פלטפורמה.

טעויות שכדאי להימנע מהן

הטעות הנפוצה ביותר היא לדחות החלטות SEO לסוף. אחריה מגיעה החלטה לא מודעת להעמיס ספריות. שלישית, התעלמות מבדיקות שימושיות, כי "נשפר אחר כך". בשטח, אחר כך מגיע יקר. עוד טעות שכיחה היא אי הבדלה בין אזורים קבועים לדינמיים, מה שמוביל ל-SSR מיותר על עמודים שיכלו להיות סטטיים. והכי חשוב, אל תדלגו על תיעוד קצר. שני עמודים של הנחיות פיתוח חוסכים שעות לכל מפתח שיצטרף בהמשך.

FAQ קצר

למי מתאים פרויקט קסטום ולא מערכת מוכנה?

לעסקים שצריכים מהירות, אבטחה וזרימות משתמש ייחודיות, או כאלה שמבקשים שליטה מלאה ב-SEO ובארכיטקטורה. אם אתם בונים נכס לטווח ארוך ורוצים להימנע מתקרות של תבניות ותוספים, זה הכיוון.

כמה זמן לוקח לבנות אתר מאפס בקוד?

בפרויקטים סטנדרטיים לוקח לרוב 6 עד 12 שבועות, תלוי בכמות התוכן והאינטגרציות. פרויקטים מורכבים יותר עשויים להגיע ל-16 עד 20 שבועות, במיוחד אם יש אזורי משתמשים או מיקוד רב-לשוני.

איך להחליט בין React ל-Next.js?

אם האתר בעיקר תוכני וחשוב דירוג אורגני, Next.js נותן יתרון ב-SSR/SSG. אם מדובר באפליקציה עם דגש כבד על אינטראקציה פנימית, React עם רנדרינג בצד הלקוח יכול להספיק, אך לרוב Next מספק איזון טוב.

האם קוד פתוח בטוח מספיק?

כן, בתנאי שמנהלים עדכונים, בודקים תלויות ומקשיחים קונפיגורציות. היתרון הוא שקוד פתוח זוכה לביקורת מתמשכת. החיסרון הוא האחריות עליכם לשמור על משמעת.

איך בונים מבנה קישורים פנימי חכם?

מתכננים היררכיה רדודה, עוגנים ברורים, ועמודי פירמידה שמעבירים Link Juice לעמודי יעד. מוודאים שאין דפים יתומים, ומנטרים קליקים כדי לחדד קישורים שמשרתים את המשתמש ואת המטרות העסקיות.

מבט אחרון קדימה

מפת דרכים מבוססת ספרינטים מעניקה מסגרת יציבה ומציאותית לבניית אתר בקוד. היא מאפשרת לנו לגעת בכל שכבה בזמן הנכון: UX, תשתית, תוכן, SEO, אבטחה, ומהירות. לאורך הדרך מתרגלים לקבל החלטות קטנות שמצטברות ליתרון תחרותי. בניית אתרים בקוד אינה רק בחירה טכנולוגית, היא בחירה ניהולית: לייצר נכס דיגיטלי שמותאם לעסק, ולא להפך.

VeloLinx & VeloWeb - פתרונות דיגיטל מתקדמים

איש קשר: רפאל (Refael) הרוש

אזור שירות: אונליין - פריסה ארצית ובינלאומית

טלפון: 050-9122133

אתרי אינטרנט:

VeloLinx - קידום אתרים ואסטרטגיית קישורים | VeloWeb - פיתוח ובניית אתרים

אודות: בית אחד לפתרונות דיגיטל מתקדמים הפועל אונליין בארץ ובעולם. אנו מרכזים מומחיות טכנולוגית ושיווקית מקיפה: VeloWeb מתמחה בפיתוח ובניית אתרים מתקדמים מבוססי קוד בהתאמה אישית מלאה (Custom), בעוד ש-VeloLinx מספקת מעטפת קידום אתרים (SEO) מקצועית ואסטרטגיות בניית קישורים עוצמתיות. יחד, אנו מעניקים לעסקים נוכחות אינטרנטית בולטת, מהירה וממוקדת תוצאות – החל משורת הקוד הראשונה ועד למקומות הראשונים בגוגל.