

עבודת גמר בתכנון ותכנות מערכות – התמחות סייבר

5 יח"ל – שאלון 883589



מערכת ReverseProxy

* **מגיש:** עידן אפלבום (ת"ז 215917378)
* **מורים מנחים:** אריק וינשטיין ושרית שוורץ
* **בית הספר:** אמי"ת גוש-דן בר-אילן
* **שנת הגשה:** 2024, תשפ"ד

# מבוא

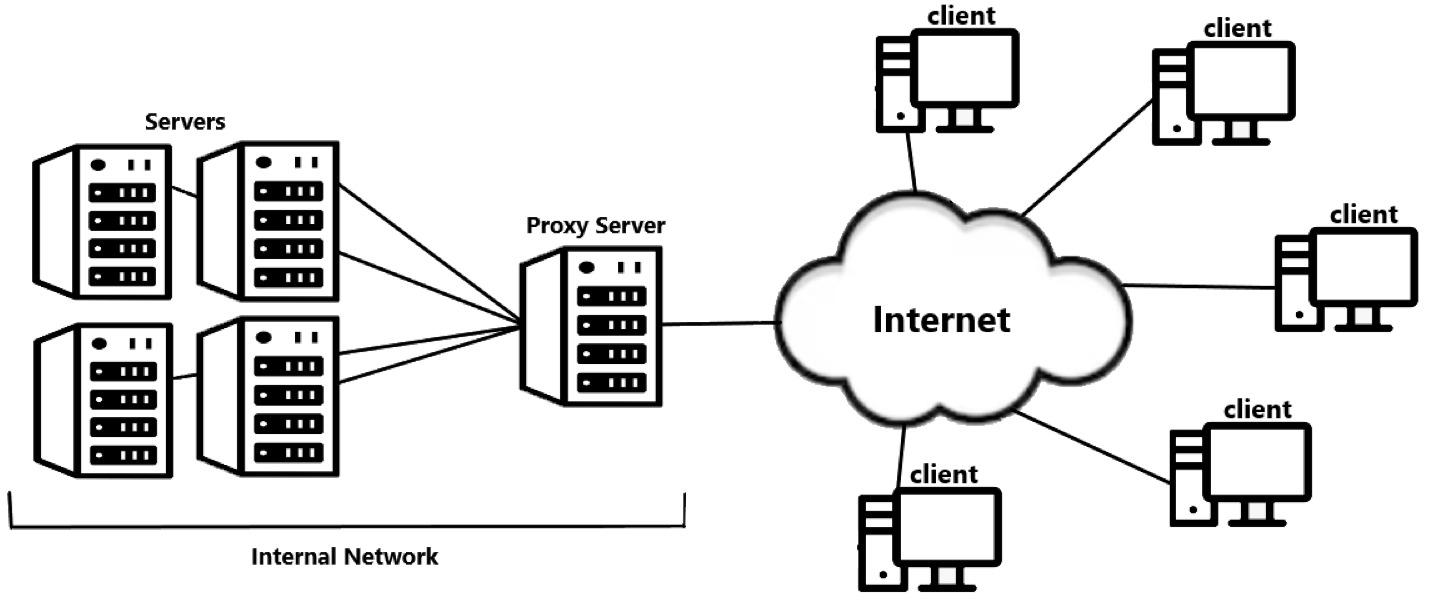
## נושא הפרויקט

RP - מערכת להגנה על שרתי web מבוססת שרת reverse proxy.

## תקציר הפרויקט

תוכנת RP היא שרת פרוקסי הפוך לניהול תקשורת רשתות שרתים לאינטרנט. שרתי פרוקסי ככלל הם שרתים המשמשים middleman בין רשת האינטרנט לשאר המכשירים ברשת מקומית (LAN). כלומר, תעבורת הרשת היוצאת והנכנסת לרשת המקומית עוברת דבר ראשון דרך הבקרה של שרת הפרוקסי. שרתי פרוקסי משמשים בעיקר לשיפור האבטחה והפרטיות על הרשת המקומית, אך גם למספר מטרות משניות כגון שיפור ביצועים עבור הגלישה באינטרנט.

בעוד שרתי פרוקסי בד"כ משמשים למשתמשים פרטיים (בצד הלקוח) ורשתות ביתיות, שרת פרוקסי הפוך משמש לרשתות שרתים של חברות מסחריות. שרת פרוקסי הפוך, כמו תוכנת RP, משמש כLoad Balancer לשרתי הרשת המסחרית. במערכת בעל רשת השרתים יוכל לבחור בין מספר אלגוריתמי ניתוב רשת על מנת למצוא את הדרך האידאלית לחלוקת בקשות הלקוחות המגיעות לרשת השרתים שלו. שירותים נוספים הניתנים ע"י הפרוקסי ההפוך הם שיפור פרטיות השרתים (כגון הסתרת הכתובת הפיזית שלהם), הסרת הצורך בהתקנת תוכנות אבטחה על כל שרת בנפרד, איסוף מידע אודות חיבורי לקוחות לשרתים ועוד.



## סיקור מצב השוק כיום

1. **שרת Apache** – שרת אינטרנט שיכול לשמש כפרוקסי הפוך. השרת מבצע מספר פעולות כגון load balancing ע"י יצירת workers שמטפלים במשתמשים, שיפור האבטחה ע"י NAT, שיפור היעילות בעזרת static data caching וחיבורי SSL המאפשרים תקשורת ברשת הפנימית ללא הצפנה.
2. **שרת NGINX** – דומה לשרת Apache, אך בא אחריו. NGINX הוא תוכנת שרת אינטרנט בקוד פתוח שיכול לשמש כשרת פרוקסי הפוך. יתרונו על Apache הוא שהוא יותר פשוט, מה שמשפר יעילות ומהירות הריצה שלו. אך כתוצאה מכך, הוא יותר מוגבל ביחס לשרת Apache.
3. המטרה של תוכנת ReverseProxy היא לנהל תקשורת נכנסת ויוצאת אל רשת השרתים שאליה היא מחוברת. RP תדאג לחלק את עומס הפניות בצורה יעילה על פני כל השרתים הרשומים במערכת. בנוסף המערכת אוספת מידע כללי אודות השימוש בשרתים ומספקת סטטיסטיקות עליהם.

## הגדרת לקוח

התוכנה מיועדת לאנשים פרטיים או ארגונים בעלי אתר אינטרנט אשר רוצים לנהל את התקשורת של שרתי הweb שלהם עם האינטרנט בצורה פשוטה ודינאמית. ReverseProxy תאפשר להם לסדר את התקשורת של השרתים שלהם ולאסוף סטטיסטיקות אודות החיבורים שלהם מול הרשת בצורה נוחה.

## תיחום הפרויקט

התוכנה תאפשר ניהול מסודר של תקשורת רשת השרתים מול הבקשות המגיעות אליה. אך התוכנה לא תיצור worker עבור כל משתמש בפני עצמו, אלא תשתמש בשרתים הנתונים. בנוסף, התוכנה לא תוכל למנוע התקפות על רשת השרתים כגון התקפות DDoS (Distributed Denial of Service). מעבר לכך, התוכנה משמשת כSPoF (Single Point of Failure), כלומר, אם המערכת מפסיקה לתת שירות מכל סיבה שהיא, כל התקשורת לא תעבוד, כל מערכת התקשורת תיפסק.

## אתגרי הפרויקט

1. **מחקר ופיתוח** – כחלק מפיתוח התוכנה, אצטרך לחקור וללמוד על נושאים חדשים כמו התקשורת בין הרכיבים שברשת, Load Balancing algorithms ועבודה מול שרתי אינטרנט.
2. **בחירת התחברות התוכנה לרשת** – יש לבחור את ספרייה להתחברות לרשת שתתאים ביותר למטרות התוכנה. צריך ספרייה שתתמוך בהתחברות למספר שרתי אינטרנט בו זמנית ובהתחברות לרשת.
3. **המצב הנוכחי בישראל** – בעקבות מלחמת חרבות ברזל שאנו נמצאים בה כרגע, עולים הרבה חששות וקשיים עקב מצב החוסר וודאות שכולנו נמצאים בו.

## תיאור המערכת

תוכנת ReverseProxy היא שרת פרוקסי הפוך המשמשת כ-Load Balancer עבור רשתות שרתי web.

התוכנה תבצע מספר שירותים:

Load Balancing – התוכנה תשאף לחלק את כניסות המשתמשים בצורה האופטימלית על מנת להפחית כמה שיותר עומס משרת בודד. התוכנה מציעה מספר אלגוריתמים שונים לחלוקת עומסים שהמשתמש יוכל לבחור מהם את האלגוריתם המתאים למצבו הפרטי.

פרטיות – כאשר נשלחת תגובה מהשרת, הלקוח רואה רק את כתובת הIP של שרת הפרוקסי, כך שאין לו כל מידע על השרת המקורי.

נוחות – כאשר יש צורך בהוספת תוכנת אבטחה, שינויים בקריפטוגרפיה או כל שינוי המשפיע על תקשורת השרתים עם האינטרנט – תוכנת RP מבטלת את הצורך לבצע שינויים בכל שרת ושרת. במקום זה, ניתן לבצע את השינויים אך ורק על שרת הפרוקסי, וכתוצאה מכך כל המערכת תושפע מהחידושים.  
התחברות דינאמית לשרתים – בתחילת התוכנית, המשתמש יתבקש להזין את כתובות השרתים אותם ירצה לחבר למערכת. המערכת מסוגלת להתמודד עם נפילות שרתים, ובמקרה כזה תפסיק לנתב אליהם בקשות ותמשיך לעבוד מול השרתים הזמינים שמולה. במצב שכל השרתים נפלו, יתבקש המשתמש להזין כתובות שרתים חלופיים או שרתים שחזרו מתיקון.

Server data logging – התוכנה תאסוף מידע אודות השרתים כגון מספר הכניסות לכל שרת, זמן ריצה באוויר וכו'.

תהליכים בתוכנה:

* קבלת בקשות התחברות מהמשתמשים ברשת
* מציאת השרת הנוכחי לתעבורת רשת ע"פ האלגוריתם הנבחר
* ניתוק שרתים לא פעילים מהמערכת
* איסוף וניתוח פעילות המשתמשים בשרתים

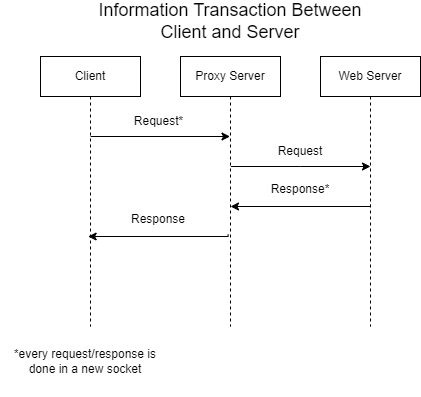
## בדיקות התוכנה

מספר בדיקות יבוצעו על מנת לבדוק שכל התהליכים במערכת עובדים כראוי, לדוגמא:

1. **טיפול בתקשורת השרת והלקוח:** בדיקה שהחיבור מאפשר לתוכנה להעביר בקשות של לקוחות ומידע אל השרתים, וששרת הפרוקסי מצליח להחזיק כמה TCP tunnels באותו הזמן.
2. **בדיקת חיבור השרתים לתוכנה:** בדיקה שהתוכנה מפנה בקשות התחברות לשרת המתאים, ודואגת לבדיקה מתמידה של מצבם לקבל התחברויות חדשות.
3. **בדיקת קריסת שרתים והתמודדות התוכנה:** בדיקה שכאשר שרת/ים נופלים המערכת מסוגלת להתמודד איתם ולהמשיך להעביר בקשות אל השרתים הפעילים.
4. **בדיקת קבלת Data Log:** בדיקה ששרת הפרוקסי RP מתעד באמינות את הסטטיסטיקות כגון מספר הכניסות לשרתים וזמן אוויר וכד', ומציג אותם במקום המתאים.

## תהליכים ראשיים במערכת

1. **קישור לקוח לשרת והעברת התקשורת ביניהם:** הלקוח שולח בקשת התחברות לאתר, שתגיע אל שרת הפרוקסי. השרת מנסה להעביר את ההודעה אל השרת הנוכחי שנבחר, ואם מצליח בכך מחזיראת התגובה שהתקבלה בחזרה אל הלקוח**.**
2. **בחירת השרת הנוכחי לקבלת הבקשות:** כל כמות זמן מוגדרת מראש או כאשר לקוח חדש מתחבר, תלוי בהגדרת המשתמש, המערכת בוחרת את השרת אליו תגיע הבקשה ע"פ האלגוריתם הנבחר ע"י המשתמש. במצב שהשרת נבחר כל פרק זמן, כל הבקשות שיגיעו עד לבחירת שרת חדש ינותבו אל אותו שרת.
3. **איסוף וניתוח פעילות המשתמשים בשרתים:** במהלך התקשורת של הלקוחות עם השרתים, התוכנה אוספת מידע כגון מספר התחברויות לכל שרת, כמות זמן שהשרת באוויר וכד, כמות זמן מאז בקשה אחרונה ומאז פעם אחרונה שקרס**.** נתונים אלה יוצגו מול מסך הבקרה של המשתמש, וישמרו לשימוש עתידי במאגר הנתונים.



A group of white rectangular objects with black text

Description automatically generated

A close up of a sign

Description automatically generatedA group of white rectangular objects with black text

Description automatically generated

פרק מבנה

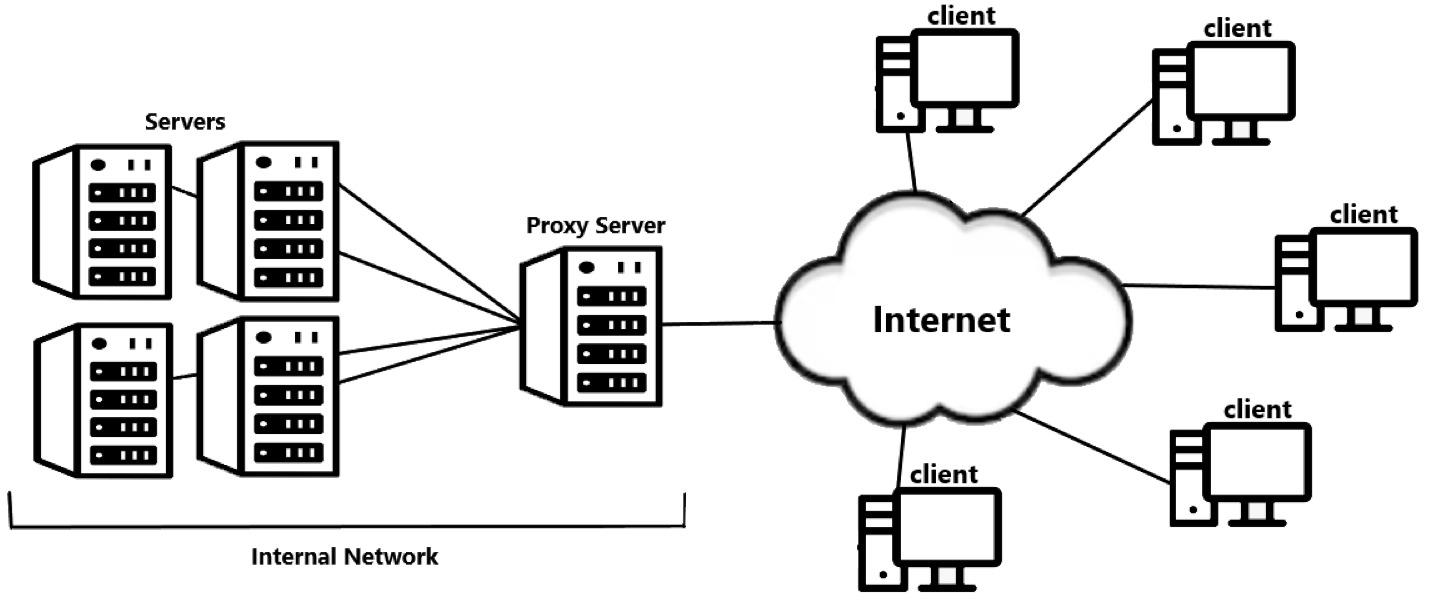
# מבנה הפרוייקט

## ארכיטקטורת המערכת

מערכת Reverse Proxy בנויה בארכיטקטורת שרת – לקוח/שרת – לקוח על בסיס תוכנת דסקטופ ושימוש בתקשורת על בסיס socket. במכלול פעילות המערכת משתתפים 3 רכיבים מרכזיים: רשת שרתי הweb, תוכנת RP והלקוחות (clients).   
רשת שרתי הweb: אלה השרתים שהמשתמש בתוכנה מעוניין לנהל. שרתים אלה נותנים שירות של דפי האינטרנט לפי בקשות בפרוטוקול HTTP. השרתים בד"כ יהיו זהים בתוכנם, אך אין תנאי זה הכרחי לפעילות המערכת בחלק מהמקרים.

תוכנת RP: מושא תיק פרוייקט זה. התוכנה מהווה בmiddle-man בין השרתים לבין הלקוחות (עליהם יוסבר בהמשך). תוכנה זו משמשת גם כשרת (מול הלקוחות ברשת) וגם כלקוח (מול רשת השרתים).

לקוחות (clients): המשתמשים ברשת האינטרנט. משתמשים אלה מעוניינים לקבל שירות (דפי אינטרנט) מהשרתים. בד"כ הבקשות מגיעות מדפדפנים. הבקשות מגיעות בפרוטוקול HTTP.

בתחילה תוכנת הפרוקסי מתחברת אל כל שרתי הweb ברשת השרתים בחיבור socket. לאחר מכן, הלקוחות המגיעים מהרשת שולחים את בקשות הHTTP שלהם אל שרת הפרוקסי, המחלק אותן בין שרתי הweb ביעילות ע"פ האלגוריתם שהוגדר ע"י בעל השרתים. לאחר שהשרתים עיבדו את הבקשות, הם שולחים את התגובה בפרוטוקול HTTP דרך socket בחזרה אל תוכנת הRP. ולבסוף, שרת הפרוקסי שולח את התגובה אל הלקוח המתאים.  
  
את התקשורת ניתן לראות בצורה יותר מסודרת בתרשים הבא:  


## אתגרי הפרוייקט

אתגר #1 – בחירת שפה לכתיבת הפרויקט.

|  |  |
| --- | --- |
| תיאור האתגר למחקר | בחירת שפה לכתיבת הפרויקט. |
| מקור מידע רלוונטי | <https://www.coursera.org/articles/python-vs-java> |
| מסקנת שלב המחקר | החלטה לשימוש בשפת python לצורך כתיבת המערכת על גבי שימוש בשפת java. Python היא שפה יותר נוחה ופשוטה לתפעול מjava, ובנוסף יש לי המון ניסיון בעבודה בשפה זו לעומת שפת java. בעזרת שפת python ניתן לכתוב קוד הרבה יותר מהר כיוון שיש פחות צורך במסגרות וכתיבת מחלקות לעומת java. |
| היישום במחקר | כתבתי את הפרויקט בשפת python. |

אתגר #2 – יצירת מערכת אחת המתפקדת גם כשרת וגם כלקוח.

|  |  |
| --- | --- |
| תיאור האתגר למחקר | יצירת מערכת אחת המתפקדת גם כשרת וגם כלקוח. |
| מקור מידע רלוונטי | <https://www.geeksforgeeks.org/socket-programming-python/> |
| מסקנת שלב המחקר | שימוש ב3 סוגי socket-ים, שניים מצד שרת ואחד מצד לקוח. מצד השרת יש סוג אחד המשמש כמוקד לקבלת לקוחות חדשים (server socket) ואחד המשמש לתקשורת עם הלקוחות המחוברים לשרת (client sockets). מצד הלקוח יש לנו עוד סוג socket המתחבר לשרת מסוים ברשת השרתים ומשמש לתקשורת עם אותו שרת. |
| היישום במחקר | היישום ReverseProxy משמש גם כשרת (מול הלקוחות ברשת) וגם כלקוח (מול רשת השרתים). |

אתגר #3 – ניהול מספר ערוצי תקשורת.

|  |  |
| --- | --- |
| תיאור האתגר למחקר | ניהול מספר ערוצי תקשורת |
| מקור מידע רלוונטי | <https://www.geeksforgeeks.org/socket-programming-multi-threading-python/> |
| מסקנת שלב המחקר | לאחר שבחנתי מספר אפשרויות כיצד לממש את התקשורת מול מספר שרתים במקביל, החלטתי להשתמש בספריית threading, שבעזרתה אני יוצר thread עבור פעילות מול כל שרת. כל הthread-ים רצים במקביל, ונמצאים תחת אותו תהליך ראשי (proccess). |
| היישום במחקר | תוכנת RP מנהלת מספר ערוצי תקשורת ע"י שימוש בthreads. |

אתגר #4 – בחירת סביבת תכנות (framework) ליישום ממשק משתמש גרפי (GUI).

|  |  |
| --- | --- |
| תיאור האתגר למחקר | בחירת סביבת תכנות (framework) ליישום ממשק משתמש גרפי (GUI). |
| מקור מידע רלוונטי | <https://www.remotely.works/blog/the-ultimate-guide-to-choosing-the-perfect-python-gui-framework> |
| מסקנת שלב המחקר | שימוש בflet כפלטפורמה (framework) לעיצוב ממשק המשתמש הגרפי. ספריית flet מאפשרת שימוש במספר רכיבים מוכנים מראש על מנת ליצור מסכים מרשימים ביופיים ותפקודם בכמות זמן קצרה למדי. ספרייה זו עדיין חדשה, ולכן אין הרבה תיעוד עליה ברשת, אך עדיין היא מכילה כלים מאוד עוצמתיים לבניית גרפיקה שחבל לוותר עליה. |
| היישום במחקר | בפרוייקט אני משתמש בספריית flet לעיצוב ממשק המשתמש. |

אתגר #5 – עבודה בזמן מלחמת חרבות ברזל

|  |  |
| --- | --- |
| תיאור האתגר למחקר | כיום, המצב בישראל קשה מאוד מכיוון שאנו נפקדנו בעל כורחנו למלחמה בכוחות הרשע של החמאס. כל המדינה יצאה משגרתה לשגרה מיוחדת, שגרה תחת מלחמה. שגרה זו מטלטלת את כל ההתנהלות הסדירה שלנו, ומציבה בפנינו אתגרים רבים. |

## שפות וסביבת עבודה

התוכנה הראשית בפרוייקט ומודוליה נכתבו בשפת python. כמו כן גם השרתים המדומים, אך דפי האינטרנט המדומים נכתבו בשפות HTML ו- JavaScript. הפרוייקט נכתב בסביבת העבודה Visual Studio Code. בחרתי בשימוש בשפה זו בגלל שזו השפה המתאימה ביותר לצורך כתיבה ועיצוב יישום desktop. שפה זו קלה להבנה ועוצמתית בכלים שהיא מאפשרת, וקיים תיעוד רב עליה ברחבי האינטרנט. הפרוייקט נכתב וניתן להרצה על גבי מערכת ההפעלה Windows (10\11). ברחבי הפרוייקט, נעשה שימוש במבנה התוכנה Network Socket, העובד על בסיס פרוטוקול התקשורת IP. Socket משמש כנקודת קצה (endpoint) בתקשורת בין מכשירים, ונחשב לאחת הצורות הנפוצות ביותר של תקשורת בין רכיבים ברשת. לפרוטוקול זה קיימת ספרייה socket המאפשרת גישה מהירה אל ממשק הNetwork Sockets במחשב.

## פרוטוקול התקשורת בין הלקוח לשרת

**HTTP**

פרוטוקול HTTP (HyperText Transfer Protocol) הוא פרוטוקול תקשורת המשמש להעברת דפי אינטרנט ושאר משאבים ברשת האינטרנט. HTTP פועל במבנה של בקשה-תגובה (request-response), בו הלקוח (בדרך כלל דפדפן) שולח בקשה לשרת, והשרת משיב בתגובה המתאימה. הבקשות והתגובות הן במבנה טקסטואלי מוגדר היטב, מה שמאפשר תקשורת ברורה ותקנית בין הצדדים.

לדוגמה, כאשר משתמש מקליד כתובת URL בדפדפן, הדפדפן שולח בקשת HTTP לשרת. הבקשה עשויה להיראות כך:

GET /index.html HTTP/1.1

Host: www.example.com

User-Agent: Mozilla/5.0

Accept: text/html

במקרה זה, הלקוח מבקש את הדף index.html מהשרת בכתובת www.example.com. השרת יגיב בבקשה זו, אם הכל תקין, בתגובה שנראית כך:

HTTP/1.1 200 OK

Content-Type: text/html

Content-Length: 1256

<!DOCTYPE html>

<html>

<head>

<title>Example Page</title>

</head>

<body>

<h1>Welcome to Example.com!</h1>

<!-- More HTML content -->

</body>

</html>

במקרה זה, השרת משיב עם קוד אישור 200 המצביע על הצלחה, ומעביר את תוכן הדף המבוקש.

קיימים מספר סוגי בקשות HTTP שונות, כאשר העיקריים הם:

GET: בקשה לקבלת משאב מהשרת. הנתונים נשלחים דרך כתובת ה-URL והבקשה אינה משנה את מצב השרת.

POST: בקשה לשליחת מידע לשרת (כגון טופס). מידע זה נשלח בגוף הבקשה ויכול לשנות את מצב השרת.

PUT: בקשה לעדכן או ליצור משאב חדש בשרת עם המידע המסופק בגוף הבקשה.

DELETE: בקשה למחיקת משאב מהשרת.

HEAD: בדומה ל-GET, אך ללא גוף התגובה. משמש לבדוק אם משאב קיים ולבדוק את הכותרות (headers) בלבד.

OPTIONS: משמשת לבדוק אילו שיטות HTTP נתמכות על ידי השרת עבור משאב מסוים.

פרוטוקול HTTP הוא הבסיס להורדת דפי אינטרנט ולתקשורת בין דפדפנים לשרתי אתרים, ומהווה חלק מרכזי בתפקוד האינטרנט כפי שאנו מכירים אותו היום.

בפרוטוקול קיימים מספר סוגי אישור (status codes) הנשלחים מהשרת אל הלקוח:

1XX - מידע

2XX - הצלחה

3XX - ניתוב למקור אחר

4XX - תקלה בצד הלקוח

5XX - תקלה בצד השרת

למשל, בתגובה לבקשה שראינו למעלה של לקוח (קבלת דף index.html), יחזיר השרת:

HTTP/1.1 **200** OK

200 – הדף התקבל בהצלחה

## תיאור מסכי המערכת

## תרשים זרימה Screen Flow Diagram

