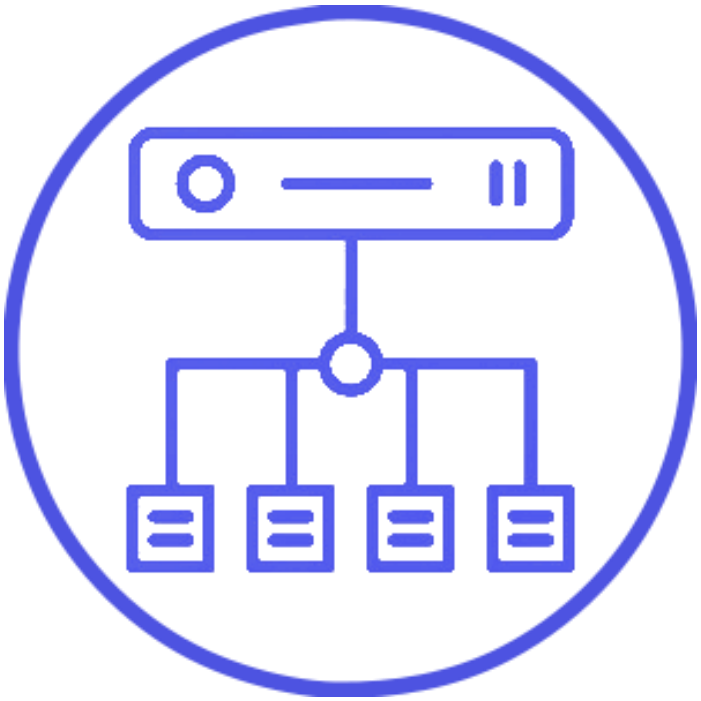
עבודת גמר בתכנון ותכנות מערכות – התמחות סייבר

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

Table of Contents

מבוא 3
נושא הפרויקט 3
תקציר הפרויקט 3
סיקור מצב השוק כיום 4
הגדרת לקוח 4
תיחום הפרויקט 5
אתגרי הפרויקט 5
תיאור המערכת 5
בדיקות התוכנה 6
תהליכים ראשיים במערכת 7
מבנה הפרויקט 11
ארכיטקטורת המערכת 11
אתגרי הפרויקט 12
תיאור האלגוריתמים המרכזיים בפרוייקט 13
שפות וסביבת עבודה 15
פרוטוקול התקשורת בין הלקוח לשרת 15
תיאור מסכי המערכת 17
תרשים זרימה Screen Flow Diagram 22
תרשים בסיסי נתונים 23
מימוש הפרויקט 25
מחלקות מיובאים/מודלים 25
מחלקות בפרויקט/מודלים 25
מודול server.py 27
קטעי קוד מיוחדים 30
מסמך בדיקות מלא 31
מדריך למשתמש 33
רשימת קבצי הפרוייקט 33
התקנת המערכת 34
רפלקציה 36
Bibliography 38

מבוא

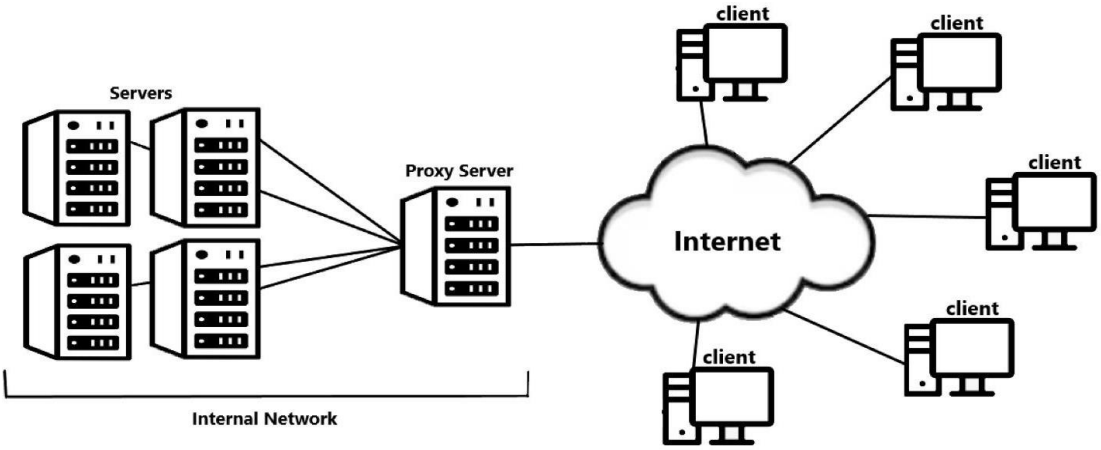
נושא הפרויקט

RP - מערכת להגנה על שרתי web מבוססת שרת ReverseProxy.

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.

1. 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

2. **מחקר ופיתוח** – כחלק מפיתוח התוכנה, אצטרך לחקור וללמוד על נושאים חדשים כמו התקשורת בין הרכיבים שברשת, Load Balancing algorithms ועבודה מול שרתי אינטרנט.
3. **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.
4. **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. (מידע אודות שרתי פרוקסי הפוכים: (© 2024 Cloudflare, Inc., 2024))

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. (מידע אודות חלוקת עומסים: 2024 ©) (Cloudflare, Inc., n.d.)

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:

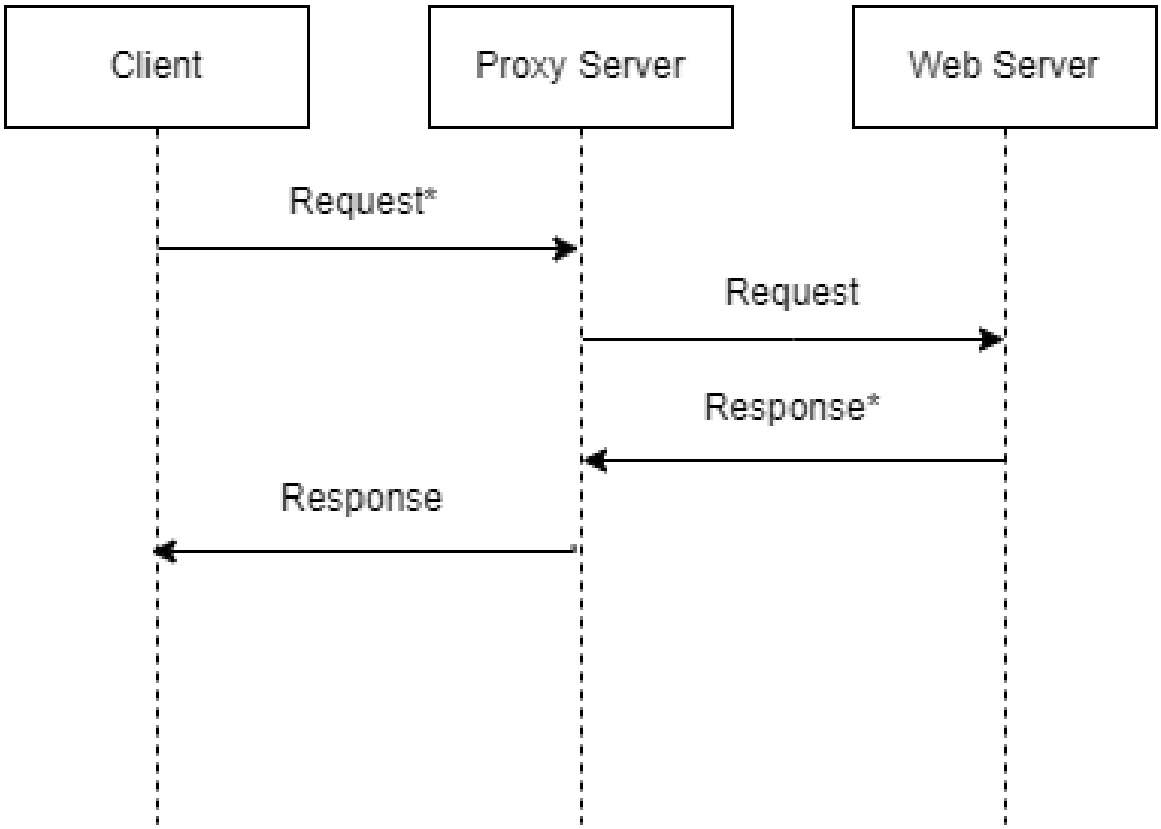
1. **טיפול בתקשורת השרת והלקוח:** בדיקה שהחיבור מאפשר לתוכנה להעביר בקשות של לקוחות ומידע אל השרתים, וששרת הפרוקסי מצליח להחזיק כמה TCP tunnels באותו הזמן.
2. **בדיקת חיבור השרתים לתוכנה:** בדיקה שהתוכנה מפנה בקשות התחברות לשרת המתאים, ודואגת לבדיקה מתמידה של מצבם לקבל התחברויות חדשות.
3. **בדיקת קריסת 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

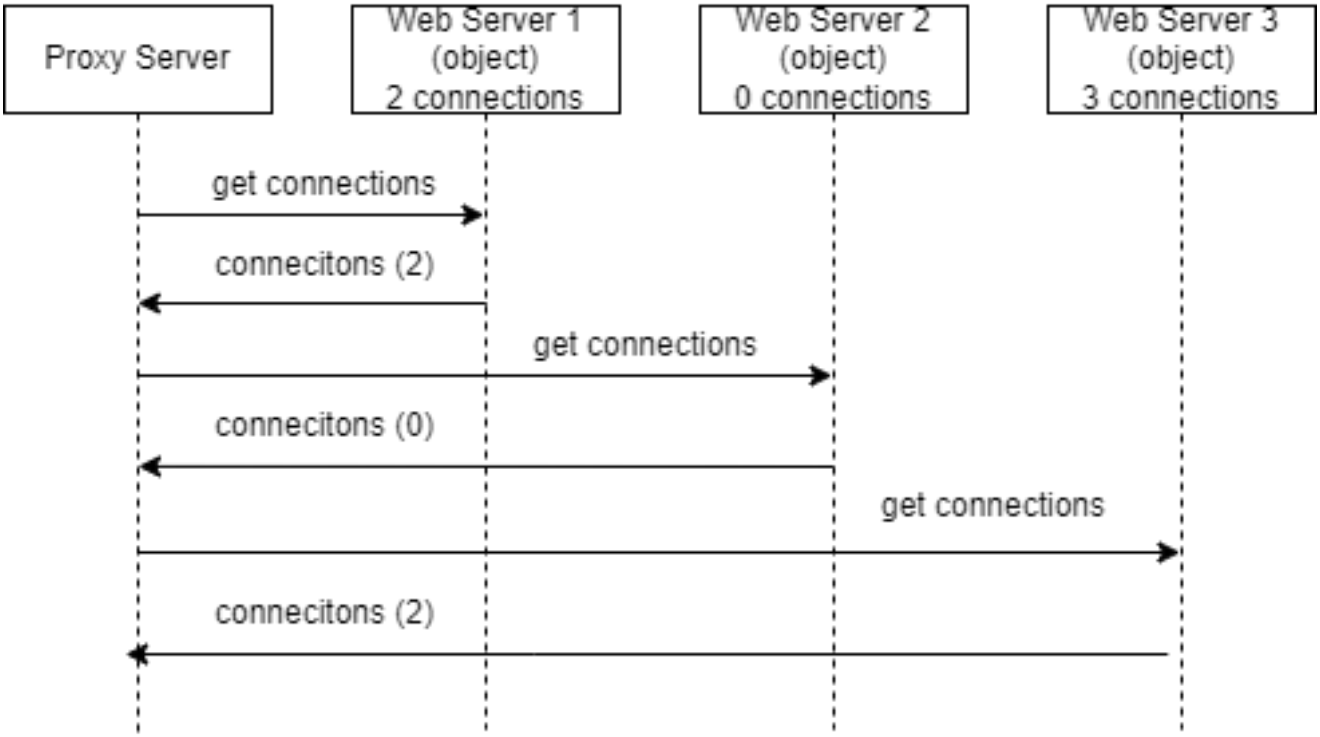
1. **קישור לקוח לשרת והעברת התקשורת ביניהם:** הלקוח שולח בקשת התחברות לאתר, שתגיע אל שרת הפרוקסי. השרת מנסה להעביר את ההודעה אל השרת הנוכחי שנבחר, ואם מצליח בכך מחזיר את התגובה שהתקבלה בחזרה אל הלקוח.
2. **בחירת השרת הנוכחי לקבלת הבקשות:** כל כמות זמן מוגדרת מראש או כאשר לקוח חדש מתחבר, תלוי בהגדרת המשתמש, 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.

Information Transaction Between Client and Server

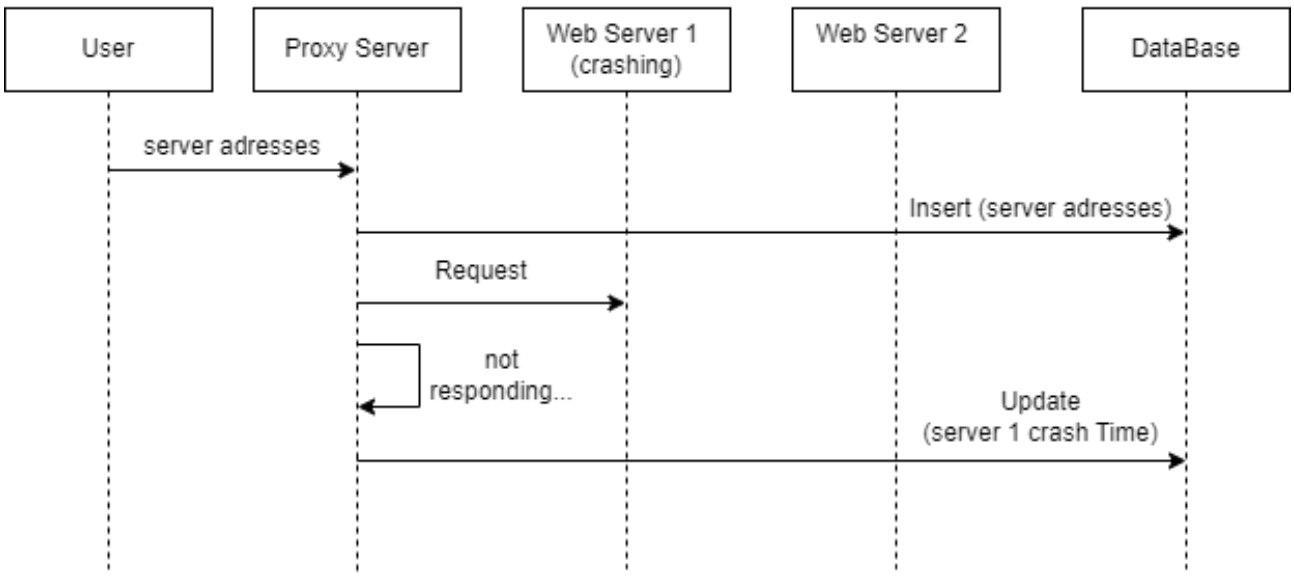


*every request/response is done in a new socket

Server Selection (leastConnections)



Data Logging (Server Adresses and Last Crashed)



פרק מבנה

מבנה הפרויקט

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).

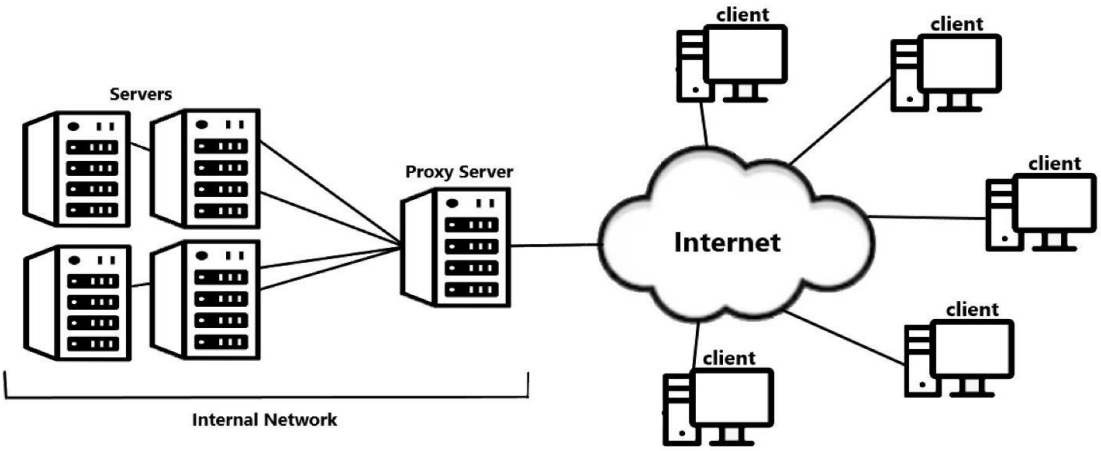
רשת שרתי הweb: אלה השרתים שהמשתמש בתוכנה מעוניין לנהל. שרתים אלה נותנים שירות של דפי האינטרנט לפי בקשות בפרוטוקול HTTP. השרתים בד"כ יהיו זהים בתוכנם, אך אין תנאי זה הכרחי לפעילות המערכת בחלק מהמקרים.

תוכנת RP: מושא תיק פרויקט 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 הפרויקט

אתגר #1 – בחירת שפה לכתיבת הפרויקט.

תיאור האתגר למחקר	בחירת שפה לכתיבת הפרויקט.
מקור מידע רלוונטי	(Staff, 2024) 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 javaמ, 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	(Verma, 2023) 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	(GeeksforGeeks, 2022) 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	(@ 2022 Remotely, n.d.) 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. (© 2024 Appveyor Systems Inc., n.d.)
היישום במחקר	בפרויקט 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.

תיאור האלגוריתמים המרכזיים בפרוייקט

הבעיה האלגוריתמית

מערכת ReverseProxy מנהלת את רשת שרתי הweb שאליה היא מחוברת. המערכת צריכה להעביר את בקשות הלקוחות ברשת אל השרתים, ולחלק אותן בצורה שהעומס יחולק כמה שיותר ביעילות. הבעיה היא כיצד התוכנית תדע לנתב את הבקשות על מנת לקבל את התוצאה הרצויה?

על השאלה הזו ניסו לענות במספר אלגוריתמים אשר גם מומשו במערכת:

1. Round Robin

אופן הפעולה: האלגוריתם מחלק את הבקשות באופן סיבובי בין השרתים. כל בקשה מופנית לשרת הבא בתור ברשימה, בלי להתחשב בעומס הנוכחי של השרתים.

יתרונות: קל ליישום ומבטיח שכל השרתים יקבלו מספר דומה של בקשות לאורך זמן.

חסרונות: לא מתחשב בעומס הנוכחי של השרתים, מה שעלול לגרום לכך ששרת עמוס יקבל בקשות נוספות.

2. Least Connections

אופן הפעולה: האלגוריתם מפנה את הבקשה לשרת עם מספר החיבורים הפעילים הנמוך ביותר. שיטה זו מתאימה במיוחד כאשר יש שוני משמעותי במשך הזמן שלוקח לכל בקשה.

יתרונות: מפחית עומס על שרתים עמוסים ומאזן בצורה טובה יותר את המשאבים.

חסרונות: יכול לדרוש יותר זמן עיבוד לבדיקה ועדכון של מספר החיבורים הפעילים בכל שרת.

3. Random

אופן הפעולה: האלגוריתם מפנה את הבקשה באופן אקראי לאחד השרתים הזמינים.

יתרונות: קל ליישום ומפזר בקשות באופן שווה פחות או יותר, במיוחד כאשר יש הרבה בקשות. בהסתברות גבוהה החלוקה תהיה שווה.

חסרונות: יכול לגרום לעומס לא מאוזן אם מספר הבקשות קטן או אם יש שוני משמעותי בזמן הטיפול בבקשות (כלומר אם אין הרבה מרחב הסתברותי).

4. IP Hashing

אופן הפעולה: האלגוריתם משתמש בערך ה-hash של כתובת ה-IP של הלקוח כדי לקבוע לאיזה שרת לשלוח את הבקשה. הערך מחושב על פי כתובת ה-IP של הלקוח ומחולק במספר השרתים, כך שתמיד יתקבל שרת ספציפי עבור כל לקוח.

יתרונות: מבטיח ש-requests מכתובת IP מסוימת ילכו תמיד לאותו שרת, מה שיכול להיות מועיל למשתמשים שדורשים מצב מתמשך (stateful).

חסרונות: לא מבטיח חלוקת עומסים מאוזנת, במיוחד אם יש לקוחות עם מספר גבוה של בקשות.

מקור מידע רלוונטי: https://www.cloudflare.com/learning/performance/types-of-/load-balancing-algorithms (© 2024 Cloudflare, Inc., n.d.)

שפות וסביבת עבודה

התוכנה הראשית בפרויקט ומודוליה נכתבו בשפת python. כמו כן גם השרתים המדומים, אך דפי האינטרנט המדומים נכתבו בשפות HTML ו-JavaScript . הפרויקט 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. הפרויקט נכתב וניתן להרצה על גבי מערכת ההפעלה Windows (11\10). ברחבי הפרויקט, 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:

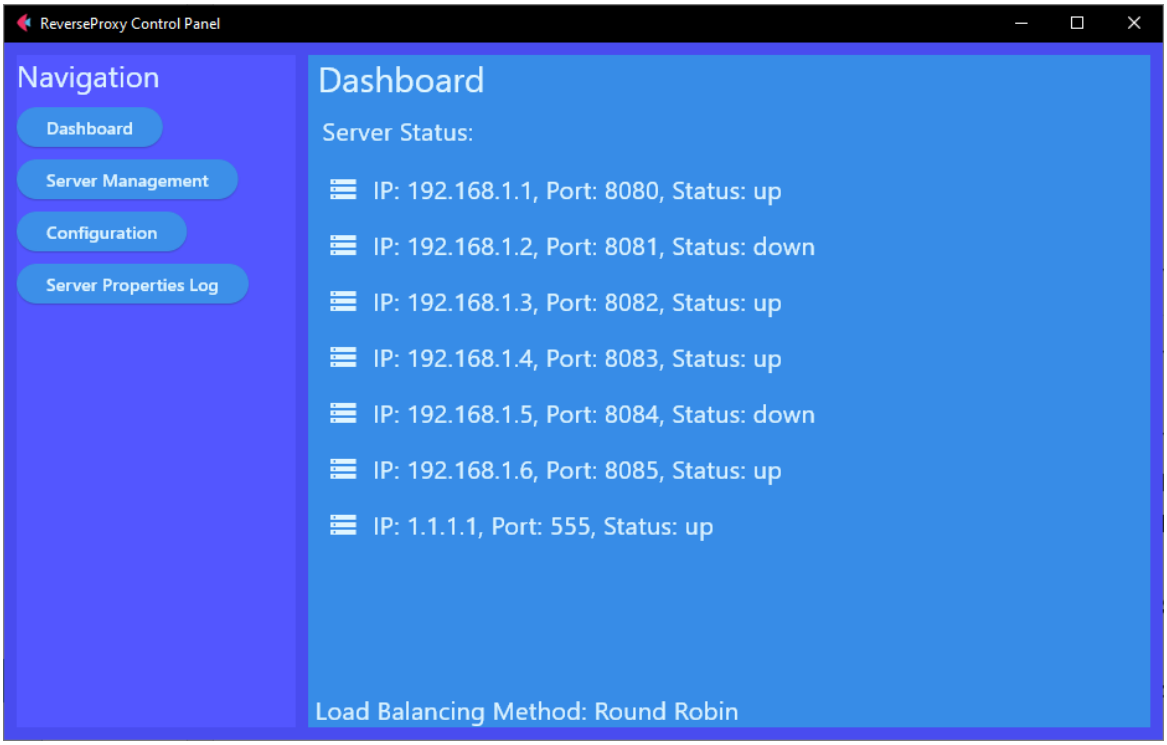
HTTP/1.1 **200** OK

200 – הדף התקבל בהצלחה

תיאור מסכי המערכת

מסך הנחיתה - DashBoard

מסך זה הוא המסך הראשון אותו המשתמש רואה כאשר הוא מפעיל את המערכת. במסך זה ניתן לראות מידע על השרתים הנוכחיים, את אלגוריתם חלוקת העומסים הנוכחי (למטה), או לעבור לאחד מהדפים האחרים בתפריט הניווט.

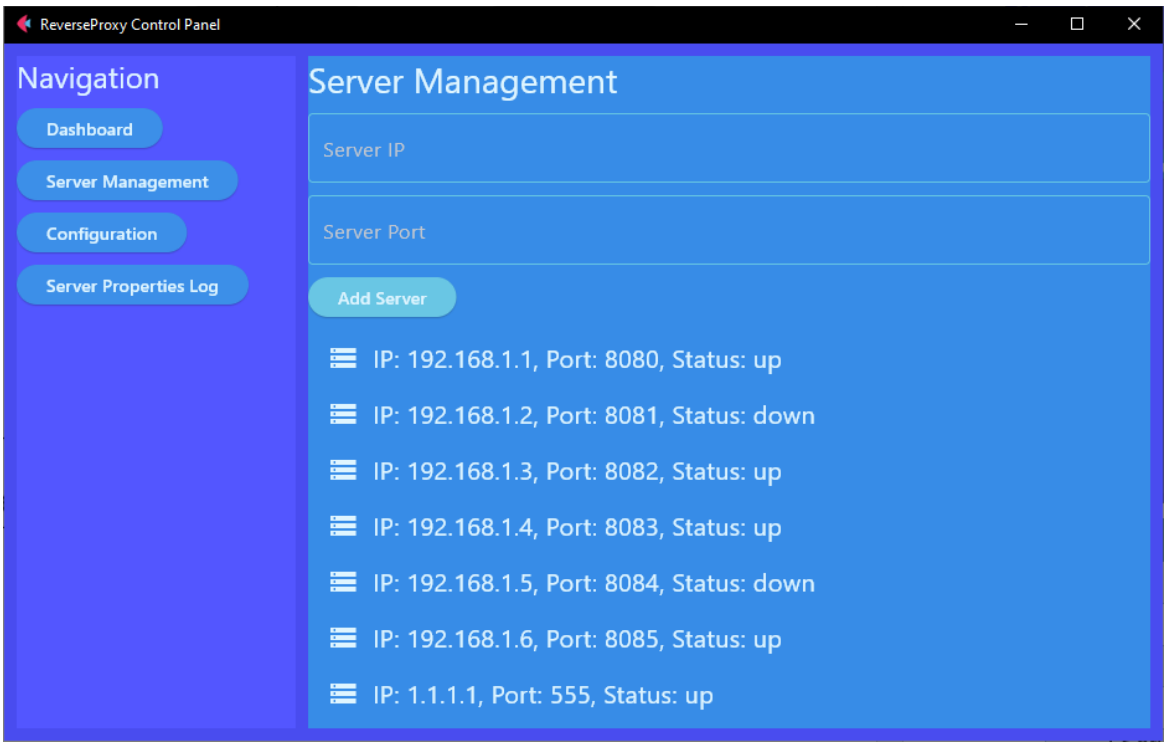


כפתור בתצוגה	פונקציונאליות
Dashboard	מעביר לדף הDashboard.
Server Management	מעביר לדף הServer Management.
Configuration	מעביר לדף הConfiguration.

Server Properties Log	מעביר לדף הServer Properties Log.

מסך ניהול השרתים – Server Management

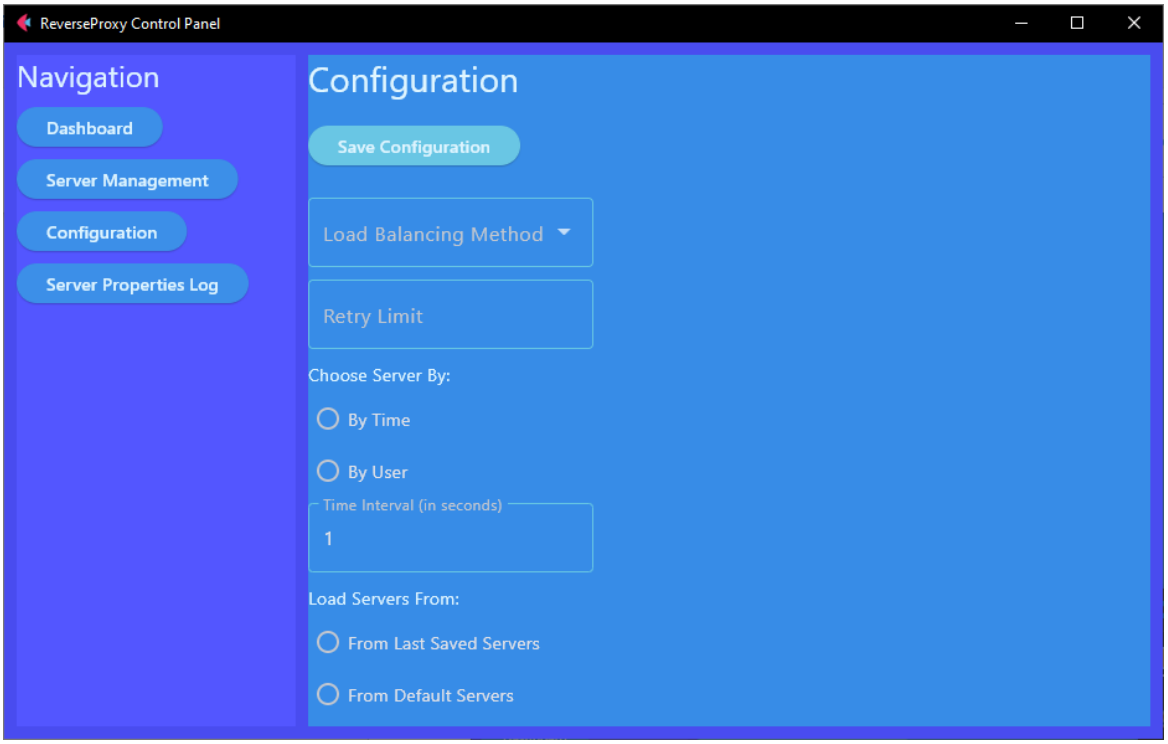
במסך זה נמצאים אפשרויות הוספה/ הסרת שרתים. ע"מ להוסיף שרת למערכת, יש לכתוב את כתובת הIP שלו ומספר הPort שלו בשדות המתאימים. ע"מ להסיר שרת מהמערכת, יש ללחוץ על אותו שרת בתפריט למטה

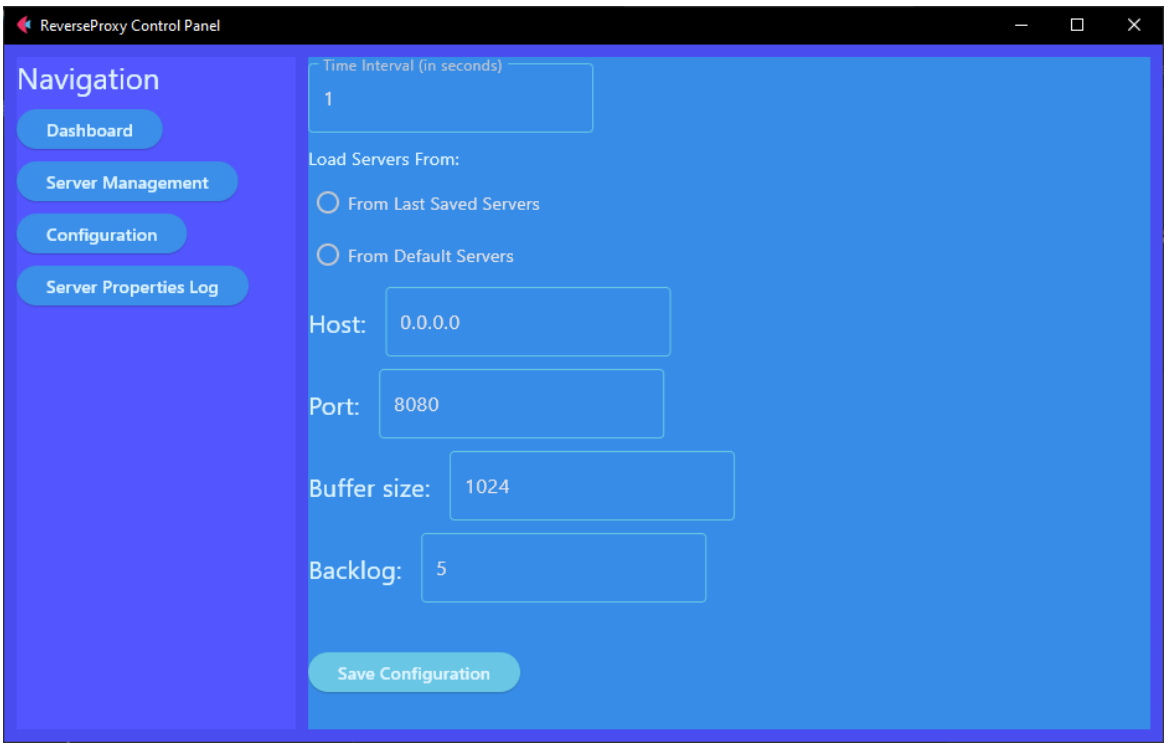


כפתור בתצוגה	פונקציונאליות
Dashboard	מעביר לדף הDashboard.
Server Management	מעביר לדף הServer Management.
Configuration	מעביר לדף הConfiguration.
Server Properties Log	מעביר לדף הServer Properties Log.
Add Server	מוסיף שרת חדש למערכת. יש לכתוב בשדות Server IP ו Server Port את כתובת האתר ומספר הפורט שלו בהתאמה.

מסך הגדרות - Configuration

במסך זה נמצאות מספר אפשרויות אשר המשתמש יכול להגדיר כדי להתאים את שרת הפרוקסי למקרה הפרטי שלו. במסך זה ניתן לבחור את אלגוריתם חלוקת העומסים מבין הרשימה, להגדיר מספר ניסיונות מחדש, אפשרות לבחור האם שרת הפניות החדש ייבחר כל כמות זמן או לפי כל לקוח שמצטרף, בחירת כמות הזמן העוברת בין כל שינוי שרת (במקרה שנבחרה האפשרות לבחירת שרת ע"פ זמן) ולשמור את השינויים.



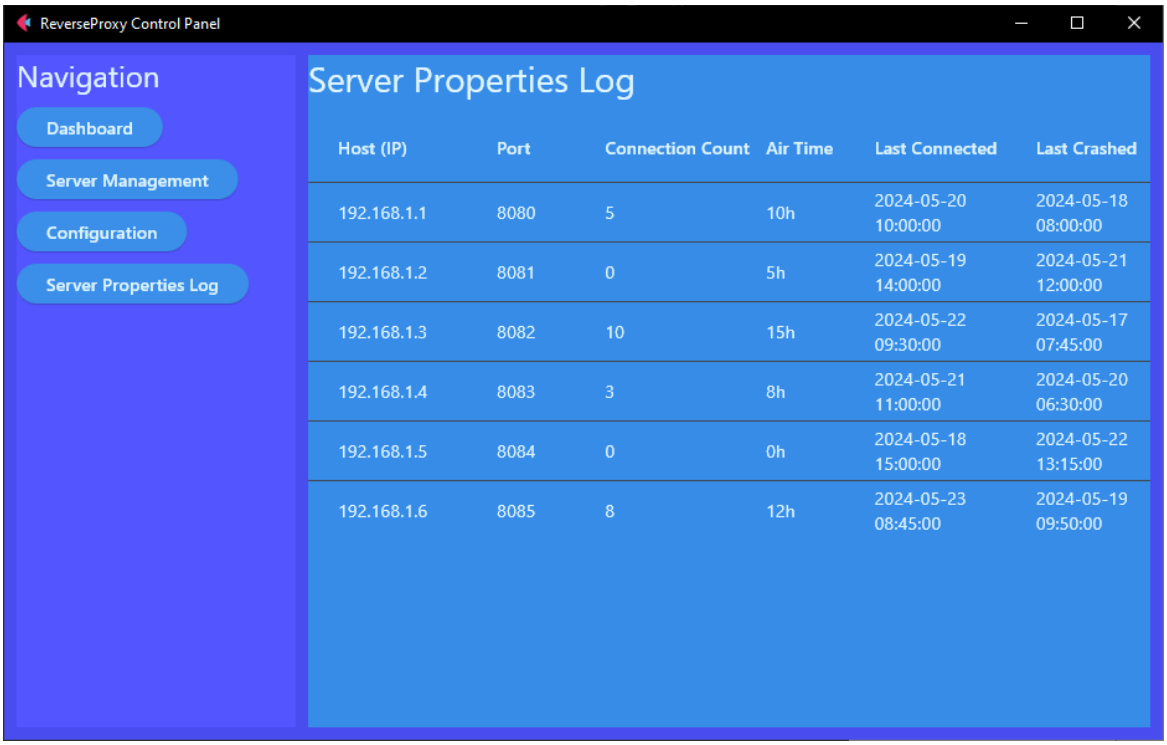


כפתור בתצוגה	פונקציונאליות
Dashboard	מעביר לדף הDashboard.
Server Management	מעביר לדף הServer Management.
Configuration	מעביר לדף הConfiguration.
Server Properties Log	מעביר לדף הServer Properties Log.

Load Balancing Method	משנה את אלגוריתם חלוקת העומסים הנוכחי. יש לבחור פריט מתוך הרשימה.
Retry Limit	אפשרות להגדיר את מספר הניסיונות המקסימאלי.
Choose Server By:	בחירה האם שרת חדש ייבחר כל כמות זמן מוגדרת או לפי כל לקוח חדש שמתחבר.
Time Interval	הגדרת כמות הזמן שבין בחירת שרת חדש (באפשרות שהוגדרה למעלה).
Load Servers From:	בחירה האם טעינת השרתים תתבצע מהשרתים שנשמרו בפעם הקודמת או מרשימה מוגדרת מראש.
Host	שינוי כתובת הIP של שרת הפרוקסי.
Port	שינוי מספר הport של שרת הפרוקסי.
Buffer Size	שינוי גודל הפקטות המתקבלות ע"י שרת הפרוקסי.
Backlog	שינוי כמות הלקוחות שיכולים לחכות ברשימת ההמתנה של שרת הפרוקסי.
Save Configuration	שמירת השינויים.

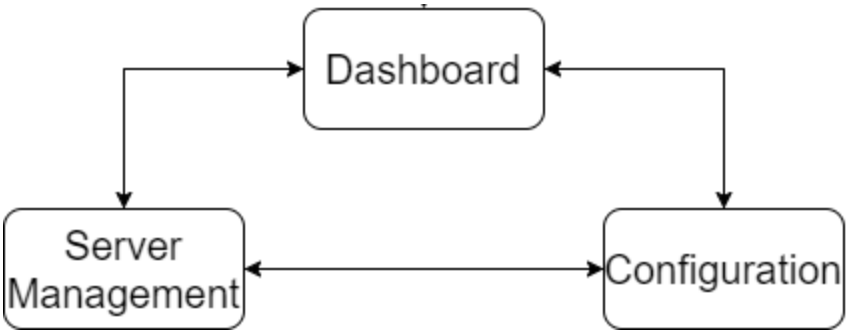
מסך מידע על השרתים – Server Properties Log

במסך זה נמצא המידע הנצבר על השרתים, כולל כתובת IP ופורט, מספר החיבורים לשרת, זמן הפעולה של השרת במערכת, הפעם האחרונה שלקוח התחבר דרך המערכת אל השרת ותאריך הקריסה האחרון שלו במערכת.



כפתור בתצוגה	פונקציונאליות
Dashboard	מעביר לדף הDashboard.
Server Management	מעביר לדף הServer Management.
Configuration	מעביר לדף הConfiguration.
Server Properties Log	מעביר לדף הServer Properties Log.

תרשים זרימה Screen Flow Diagram



תרשים בסיסי נתונים

serverData
Host
Port
Connections
Airtime
LastConnected
LastCrashed

שם השדה	סוג הערך	הסבר
Host	Text	כתובת הIP של השרת.
Port	Integer	מספר הport של השרת.
Connections	Integer	מספר המשתמשים שהתחברו אל השרת. מיועד לסטטיסטיקות, לא שימוש ישיר.
Airtime	Text	זמן הריצה של השרת. מיועד לסטטיסטיקות, לא שימוש ישיר.
LastConnected	Text	התאריך והשעה של ההתחברות האחרונה של לקוח לשרת. מיועד לסטטיסטיקות, לא שימוש ישיר.
LastCrashed	Text	התאריך והשעה של מקרה הקריסה האחרון של השרת. מיועד לסטטיסטיקות, לא שימוש ישיר.

מימוש הפרויקט

מימוש הפרויקט

מודלים/מחלקות מיובאים

שם המחלקה	תקפיד המחלקה
Socket	המחלקה מספקת גישה לממשק הרשת הנמוך לצורך יצירת וחיבור סוקטי תקשורת.
Select	המחלקה מאפשרת מעקב אחרי מספר חיבורי socket כדי לראות מתי הם מוכנים לקריאה או כתיבה.
Threading	המחלקה מספקת ממשק ליצירה וניהול של תהליכונים (threads) לפעולות במקביל.
Re	המחלקה מאפשרת עבודה עם מסננים לצורך חיפוש והתאמה של טקסט.
Random	המחלקה מספקת פונקציות ליצירת מספרים אקראיים ועבודה עם מערכים בצורה אקראית.
Logging	המחלקה מספקת כלים לרישום לוגים ומעקב אחרי אירועים והרשומות בתוכנית.
Time	המחלקה מספקת פונקציות לעבודה עם תאריכים ושעות.
Hashlib	המחלקה מספקת אלגוריתמים להפקת חותמות דיגיטליות (hash) לצורך אבטחה ואימות.
Sqlite3	המחלקה מאפשרת עבודה עם מסדי נתונים של SQLite מתוך קוד פייטון.

מודלים/מחלקות בפרויקט

מודול ProxyServer.py –

מחלקת LoadBalancer (יוצרת אובייקט שאחראי על חלוקת העומסים בproxy)

שם התכונה	תפקיד ושימוש
Self.**servers**	מכיל את רשימת השרתים המחוברים לשרת הproxy.
Self.**currentServerIndex**	מכיל את המיקום ברשימה של השרת שנבחר פעם קודמת (באלגוריתמים כגון roundRobbin).

שם הפעולה	טענת כניסה	טענת יציאה
__init__	Self, servers	הפעולה מאתחלת את מחלקת ה-LoadBalancer עם רשימת שרתים ומציינת את מיקום השרת הנוכחי. החזרה: אין החזרה (זוהי פונקציית אתחול).
chooseServer	Self, method	הפעולה בוחרת שרת על פי האלגוריתם שהוגדר, או ברירת המחדל אם שם האלגוריתם לא תקין. אם אין שרתים זמינים, הפעולה מעדכנת את רשימת השרתים. החזרה: מחזירה אובייקט שרת (Server) שנבחר.
updateServerList	Self, serverList	הפעולה מעדכנת את רשימת השרתים ומאפסת את אינדקס השרת הנוכחי. החזרה: אין החזרה (מעודכנת רק משתנים פנימיים).
getServerForClient	Self, client	הפעולה בוחרת שרת עבור לקוח נתון בהתבסס על פונקציית hashing. אם אין שרתים זמינים, מעדכנת את רשימת השרתים. החזרה: מחזירה אובייקט שרת (Server) הנבחר ע"פ האלגוריתם.
roundRobin	Self	הפעולה מחזירה שרת לפי שיטת הסיבוב (round-robin), מעדכנת את אינדקס השרת הנוכחי. החזרה: מחזירה אובייקט שרת (Server) הנבחר ע"פ האלגוריתם.
leastConnections	Self	הפעולה בוחרת את השרת עם מספר החיבורים הנמוך ביותר. החזרה: מחזירה אובייקט שרת (Server) הנבחר ע"פ האלגוריתם.
random	Self	הפעולה בוחרת שרת אקראי מתוך הרשימה.

		החזרה: מחזירה אובייקט שרת (Server) הנבחר ע"פ האלגוריתם.

מודול server.py

מחלקת BaseConnection (מחלקת האב למחלקות Server ו Client, מכילה פונקציונאליות בסיסית לאובייקט בעל תקשורת)

שם התכונה	תפקיד ושימוש
Self.**host**	כתובת הIP של האובייקט. בעזרתו ניתן להתחבר אל האובייקט.
Self.**port**	מספר הport של האובייקט. בעזרתו ניתן להתחבר אל האובייקט.
Self.**name**	שם מזהה האובייקט. נועד לlogging ותיעוד, חסר פונקציונאליות.
Self.**socket**	מכיל Socket שדרכו ניתן לתקשר עם האובייקט.
Self.**initTime**	מכיל את זמן היווצרות האובייקט במחלקה. נועד למטרות תיעוד, חסר פונקציונאליות.
Self.**type**	סוג האובייקט (Server, Client). נועד לlogging ותיעוד, חסר פונקציונאליות.

שם הפעולה	טענת כניסה	טענת יציאה
__init__	Self, host, port	הפעולה מאתחלת את האובייקט עם כתובת ה-Host, פורט, יוצרת סוקט, וקובעת את זמן האתחול. החזרה: אין החזרה (פונקציית אתחול).
connectToServer	Self	הפעולה מתחברת לשרת בכתובת ה-Host ובפורט שהוגדרו באתחול, ומדווחת ללוג על החיבור שנוצר. החזרה: אין החזרה.
closeConnection	Self	הפעולה סוגרת את החיבור לסוקט ומדווחת ללוג על סיום החיבור. החזרה: אין החזרה.

sendData	Self, data	הפעולה שולחת את המידע הנתון (data) דרך הסוקט לשרת. במקרה של שגיאת חיבור, מחזירה ערך ERROR. החזרה: אין החזרה במקרה של הצלחה; מחזירה ERROR במקרה של שגיאת חיבור.
reciveData	Self	הפעולה מקבלת נתונים מהשרת דרך הסוקט. במקרה של שגיאת חיבור, מחזירה ערך ERROR. החזרה: מחזירה את הנתונים שהתקבלו כסוג bytes או ERROR במקרה של שגיאת חיבור.
getTime	Self	הפעולה מחזירה את הזמן הנוכחי בפורמט המוגדר ב-TIME_FORMAT. החזרה: מחזירה את הזמן הנוכחי כ- int בפורמט המוגדר.

מחלקת Client (מחלקה זו מייצגת לקוח המגיע מהרשת. היא מאפשרת ניהול של הפעולות הדרושות מול אותו לקוח. מחלקה זו יורשת כמעט את כל כולה ממחלקת האב BaseConnection)

מחלקת Server (מחלקה זו מייצגת שרת web המחובר למערכת. היא מאפשרת ניהול של הפעולות הדרושות מול אותו שרת. מחלקה זו יורשת ממחלקת האב BaseConnectio)

שם התכונה	תפקיד ושימוש
Self.**clientList**	מכיל את רשימת הלקוחות לטיפול ע"י השרת.
Self.**thread**	מכיל את הThread של השרת. תהליכון זה מאפשר לשרת לרוץ במקביל לתוכנית הראשית.
Self.**runThread**	מחזיק ערך בוליאני האם התהליכון ימשיך לרוץ. מחזיק דגל.
Self.**clientCount**	מכיל את מספר הלקוחות שהתחברו לשרת. נועד למטרות תיעוד, חסר פונקציונאליות.
Self.**lastRequestTime**	מכיל את זמן הבקשה האחרונה לשרת. נועד למטרות תיעוד, חסר פונקציונאליות.

Self.**lastCrashTime**	מכיל את זמן הקריסה האחרונה של השרת. נועד למטרות תיעוד, חסר פונקציונאליות.

שם הפעולה	טענת כניסה	טענת יציאה
__init__	Self, host, port	הפעולה מאתחלת את אובייקט השרת עם פרטי ה-Host והפורט, מפעילה את מחלקת האב, מגדירה רשימת לקוחות, יוצרת תהליכון, ומתחילה את זמן החיבור האחרון והזמן האחרון שבו השרת קרס. החזרה: אין החזרה (זוהי פונקציית אתחול).
connectToServer	Self	הפעולה מנסה להתחבר לשרת. במקרה של שגיאת חיבור, סוגרת את החיבור, רושמת בלוג את זמן הקריסה, ומחזירה False. החזרה: מחזירה True במקרה של חיבור מוצלח, ו-False במקרה של כישלון.
startThread	Self	הפעולה מתחילה את התהליכון לחיפוש לקוחות אם הוא לא רץ כבר. החזרה: אין החזרה.
lookForClients	Self	הפעולה מאזינה לחיבורים מלקוחות, מטפלת בבקשות מלקוחות קיימים, ומעדכנת את מצב החיבור. אם השרת לא מצליח להתחבר, עוצרת את התהליכון. החזרה: אין החזרה.
sendRequest	Self, msg	הפעולה שולחת הודעה לשרת. במקרה של שגיאת חיבור, סוגרת את החיבור, רושמת בלוג את זמן הקריסה, ומחזירה False. החזרה: מחזירה True במקרה של שליחה מוצלחת, ו-False במקרה של כישלון.
reciveResponse	Self	הפעולה מקבלת תגובה מהשרת בהמשכים עד סוף הנתונים או שגיאת חיבור. החזרה: מחזירה את התגובה כסוג bytes, או ERROR במקרה של שגיאת חיבור.

handleRequest	Self, client	הפעולה מקבלת בקשה מלקוח, שולחת אותה לשרת, מקבלת את התגובה, ושולחת אותה בחזרה ללקוח. החזרה: אין החזרה.
insertClient	Self, newClient	הפעולה מוסיפה לקוח חדש לרשימת הלקוחות של השרת. החזרה: אין החזרה.
removeClient	Self, client	הפעולה מסירה לקוח מרשימת הלקוחות של השרת אם הוא קיים ברשימה. החזרה: אין החזרה.

קטעי קוד מיוחדים

```
def checkServerProps(serverProps: str) -> zip:
    serverProps = "^" + serverProps
    IPv4Format = "[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}"
    hosts = re.findall(IPv4Format, serverProps)
    for host in set(hosts):
        serverProps = serverProps.replace(host, ".")
    ports = re.findall("[^0-9][0-9]{1,5}", serverProps)
    ports = [x[1:] for x in ports]
    return zip(hosts, ports)
```

מטרת קטע הקוד שלמעלה היא לחלץ זוגות של (host, port) מתוך מחרוזת נתונה. היופי בפעולה זו היא הדינאמיות המרשימה שהיא מציעה בנוגע לאופן שמירת הזוגות בתוך המחרוזת. ניתן לשמור אותם עם הפרדה של ; למשל, או הפרדה של ירידת שורה, או אפילו במקרים מסויימים ללא הפרדה בכלל! (במקרים שכל התווים מלאים ע"פ הפורמט)

ע"י שימוש בספריית RegEx (re), הפעולה מסוגלת לבצע חיפושים על קטעי תווים ספציפיים במחרוזת המקיימים תנאים מסויימים. למשל, הפעולה מחפשת את כל כתובות הIP ע"פ הפורמט המוגדר במשתנה IPv4Format, ומתעלמת מכל שאר המחרוזת הלא רלוונטית.

לאחר מכן, הפעולה מוציאה את כל כתובות הIP מהמחרוזת, ומתחילה לחפש כתובות port. בסופו של דבר, הפעולה מחזירה את כל הזוגות שנמצאו לפי סדר הימצאותם במחרוזת.

מסמך בדיקות מלא

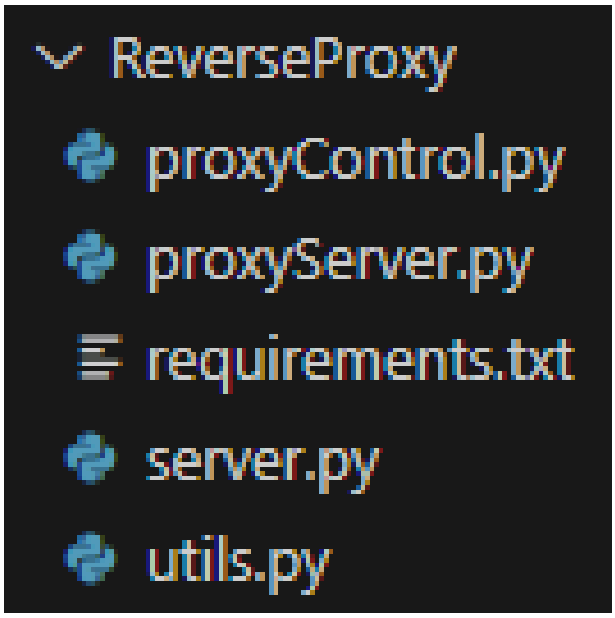
מטרת הבדיקה	אופן הביצוע	תוצאות הבדיקה	פתרון בעיות
בדיקות טיפול בתקשורת השרת והלקוח	הרצה של כמה לקוחות, ובקשת התחברות וקבלת דפי אינטרנט דרך כל לקוח.	הדפים הוצגו בפני הלקוחות, כלומר המערכת הצליחה להחזיק כמה ערוצי תקשורת ולנהל אותם באותו הזמן.	אין
בדיקות חיבור השרתים לתוכנה	הרצה של כמה לקוחות, ובקשת התחברות וקבלת דפי אינטרנט דרך כל לקוח.	רוב הדפים שהוצגו בפני הלקוחות היו נכונים, אך חלקם קיבלו חלק משרת אחד וחלק משרת בעל תוכן שונה.	שימוש באלגוריתמים המקשרים בין לקוח לשרת לפי כתובת הלקוח (IP hashing) במצב שמחוברים למערכת שרתים בעלי תוכן שונה.
בדיקת קריסות שרתים והתמודדות המערכת	הרצה של כמה שרתים, חיבורם למערכת והפסקת פעילותם במכוון ע"מ לדמות נפילה של שרת.	לרוב המערכת הוציאה בהצלחה את השרתים המנותקים, אך באלגוריתם ה roundRobin, המערכת ניסתה לפנות לתא לא קיים ברשימת השרתים – דבר שהוביל לקריסתה.	הוספה על פונקציית הוצאת שרתים, איפוס המיקום הנוכחי של השרת ברשימה. ובנוסף הוספת try…except סביב קבלת השרת הנוכחי ע"מ לתפוס ולנטרל נפילות אלה.
בדיקת קבלת Data Log	דימוי פעילות רגילה במערכת, ובסופה בדיקת מאגר המידע.	המידע תועד באמינות בבסיס הנתונים.	אין

מדריך למשתמש

מדריך למשתמש

רשימת קבצי הפרוייקט

עץ קבצי המערכת:



שם הקובץ	תיאור
proxyControl.py	הממשק הגרפי של המערכת.
proxyServer.py	קובץ המכיל את כל הפונקציות הקשורות בשרת הפרוקסי עצמו. (Load Balancer וה- proxy server). כמו כן תיעוד המידע בDB.
requirements.txt	קובץ ההתקנות עבור pip.
utils.py	קובץ המכיל פעולות עזר.

התקנת המערכת

צד שרת

דרישות: פייתון מגרסה 3.11 ומעלה ומנהל החבילות pip.

מומלץ להשתמש במכונה הוירטואלית של פייתון על מנת להריץ את השרת, אם הוא לא מותקן הריצו בשורת הפקודות pip install venv. להלן ההוראות להרצת השרת:

1. פתחו את שורת הפקודות וגשו לתיקיית הפרויקט.
2. הריצו את הפקודה python -m venv venv. תפתח תיקייה חדשה בשם venv.
3. הפעילו את המכונה הוירטואלית על ידי הרצה של activate/Scripts/venv. תוכלו לראות שזה עבד על ידי כיתוב (venv) בכל שורה בשורת הפקודות.
4. התקינו את הספריות הדרושות. ניתן לעשות זאת בקלות על ידי הפקודה pip install -r requirements.txt.
5. הריצו את הקובץ proxyControl.py.

צד לקוח

דרישות: דפדפן תקין.

פתחו את הדפדפן והקלידו בשורת הURL: **127.0.0.1**:8080 (או את כתובת המכונה אשר עליה רץ שרת הפרוקסי).

רפלקציה

רפלקציה

תהליך העבודה על הפרויקט

עבודה על פרויקט ReverseProxy הייתה מסע מרתק ומאתגר. התחלתי בהבנת הדרישות והגדרת מטרות הפרויקט. במהלך התכנון, היה עליי לחקור ולהבין לעומק את נושא ה- Load Balancing והפרוקסי ההפוך, ולבחור את הכלים המתאימים ליישום המערכת. אחת ההצלחות הייתה בחירת שפת התכנות Python וסביבת הפיתוח Visual Studio Code, מה שאיפשר לי לפתח את המערכת בצורה מהירה ויעילה. אך יחד עם ההצלחות היו גם אתגרים משמעותיים, כמו ניהול מספר ערוצי תקשורת במקביל והתמודדות עם קריסות שרתים.

תהליך הלמידה שעברתי

במהלך הפרויקט למדתי הרבה על תחום ה-Load Balancing, כולל מחקר מעמיק על אלגוריתמים שונים כמו Round Robin, Least Connections, Random ו-IP Hashing. נאלצתי להתמודד עם אתגרים טכניים כמו תכנות ב-socket והבנה מעמיקה של Threading ב-Python. מעבר לכך, למדתי גם על בניית ממשק משתמש גרפי בעזרת ספריית flet.

כלים שאקח להמשך

מהפרויקט אני לוקח איתי ידע מעמיק בתחום התקשורת ברשת, פרוקסי הפוך ו- Load Balancing. בנוסף, רכשתי מיומנויות חדשות בתכנות בשפת Python, עבודה עם Sockets ו-Threading, ופיתוח ממשקי משתמש גרפיים. כלים אלו יהיו מאוד שימושיים בפרויקטים עתידיים בתחום פיתוח מערכות ותכנות מערכות מורכבות.

תובנות מהתהליך

התהליך לימד אותי את החשיבות של מחקר מעמיק והבנת הבעיה לפני התחלת הפיתוח. בנוסף, הבנתי את החשיבות של תכנון נכון והתמודדות עם אתגרים טכניים בצורה שיטתית. השיתוף והעזרה ממורים מנחים ומקצוענים בתחום הייתה חיונית להצלחת הפרויקט. כמו כן, למדתי את הערך של למידת עמיתים ושיתוף מידע, מה שהעשיר את הידע שלי וסייע בפתרון בעיות.

תודות

ברצוני להודות למורים המנחים, אריק וינשטיין ושרית שוורץ, על הליווי המקצועי והעזרה לאורך כל התהליך. תודה גם לחברי הכיתה ולעמיתים שסייעו ונתנו משוב מועיל. בנוסף, תודה למשפחתי על התמיכה וההבנה במהלך תקופת העבודה על הפרויקט.

ביבליוגרפיה

Bibliography

@ 2022 Remotely. (n.d.). *The Ultimate Guide to Choosing the Perfect Python GUI Framework*. Retrieved from Remotely: https://www.remotely.works/blog/the-ultimate-guide-to-choosing-the-perfect-python-gui-framework

© 2024 Appveyor Systems Inc. (n.d.). *Tutorials*. Retrieved from flet: https://flet.dev/docs/tutorials/

© 2024 Cloudflare, Inc. (2024). *What is a reverse proxy? | Proxy servers explained*. Retrieved from Cloudflare: https://www.cloudflare.com/learning/cdn/glossary/reverse-proxy/

© 2024 Cloudflare, Inc. (n.d.). *Types of load balancing algorithms*. Retrieved from Cloudflare: https://www.cloudflare.com/learning/performance/types-of-load-balancing-algorithms/

© 2024 Cloudflare, Inc. (n.d.). *What is load balancing? | How load balancers work*. Retrieved from CloudFlare: https://www.cloudflare.com/learning/performance/what-is-load-balancing/

GeeksforGeeks. (2022, Jul 14). *Socket Programming with Multi-threading in Python*. Retrieved from Geeks for Geeks: https://www.geeksforgeeks.org/socket-programming-multi-threading-python/

Staff, C. (2024, April 4). *Python vs. Java: Which Should I Learn?* Retrieved from Coursera: https://www.coursera.org/articles/python-vs-java

Verma, K. (2023, Feb 28). *Socket Programming in Python*. Retrieved from Geeks for Geeks: https://www.geeksforgeeks.org/socket-programming-python/