**בדיקות קיבול ועומס**

במסמך זה נגדיר את המדדים לקיבול ועומס במערכת, נתכנן את הבדיקות המתאימות וננתח תרחישי הרצה שונים.

תוכן עניינים –

תמהיל תרחישי שימוש ........................................................................................................... 2

תכנון הבדיקות בקווים כללים ................................................................................................... 3

**תרחישי בדיקה**

בדיקות עומס .....................................................................................................................-7 …4

בדיקות עומס ......................................................................................................................... 8

**תמהיל שימוש במערכת**

המטרה SLO- מערכת המסחר שלנו צפויה לשמש לקוחות שונים, כאשר כל אחד מהם ראשי לבצע פעולות מגוונות במערכת, למשל רכישות מוצרים, פתיחת חנות, ניהול חנות בכל התחומים, צפייה בהיסטוריית רכישות וכו'.

מכיוון שכל פעולה שכזו דורשת הרשאה שונה, עלינו לדאוג שהמערכת תתמוך בכל סוגי המשתמשים ובפעילות מקבילית שלהם.

בסיס הנתונים צריך להיות קוהרנטי עם המידע במערכת ולתמוך בתחזוקה של מידע רב מאוד ובמספר גדול של אובייקטים, על המערכת להמשיך לתת שירותים למשתמשים.

העומס הצפוי להיות על המערכת היא כ- 100 משמשים מנויים באותו הזמן, נראה שזמינות האתר תהיה בכל שעות היום ועבור כל משתמש.

על מנת למדוד את העמידה במטרה נגדיר מדדי SLI קונקרטיים-

* התמודדות עם 100 בקשות בו זמנית כך שזמן התגובה הממוצע לבקשה יהיה לכל היותר שנייה אחת. דוגמא לבקשות – הרשמה, פתיחת חנות, וכו.
* תמיכה בכמות רבה של מידע, נגדיר – 1000 חנויות, ממוצע של 1000 מוצרים בכל חנות, כאשר במערכת 1000 משתמשים רשומים ובהיסטוריה של עד 1,000,000 רכישות. על בסיס הנתונים להכיל את המידע ולתחזק אותו בהתאם לפעילות המערכת, על המערכת לא לקרוס ולהמשיך לתת שירותים למשתמשים. נגדיר את העמידה במדדים SLA להיות ב 100 אחוז מהמקרים.
* תמיכה ב 1,000 מבקרים במערכת בכל רגע נתון העמידות של מדדי SLA = 100 % .
* המערכת תהיה עמידה באירועים לא צפויים כמו נפילות תקשורת או קשר לרכיבים שונים.

**תכנון הבדיקות – קווים כללים**

* תמיכה ב Lazy Load על מנת לעמוד במדדי הבדיקות מבחינת מידע רב של המערכת. כך גם אם בבסיס הנתונים יש מידע רב של מאות אלפי אובייקטים, לא יהיה עומס ברמה כזאת על הזיכרון הראשי על מנת שנוכל לתת שירות למשתמשים בזמן סביר.   
  דוגמא שממומשת במערכת בטעינה עצלה –

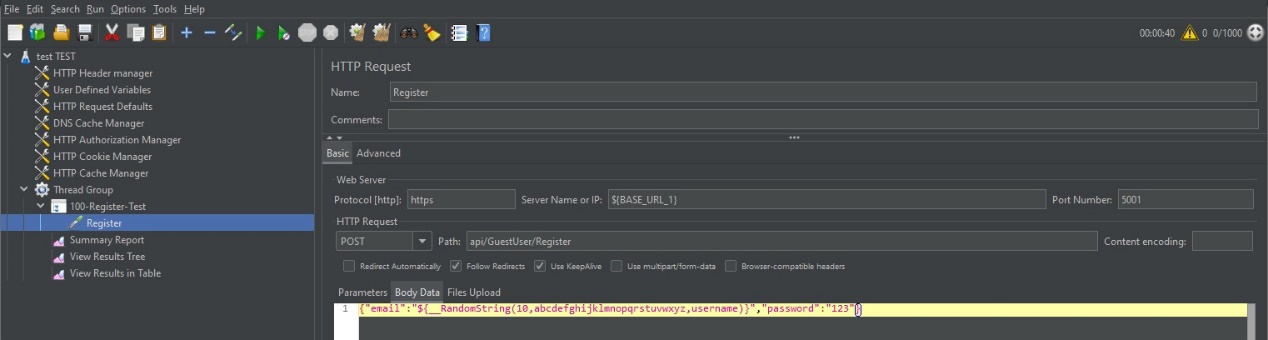
1. לא נטענים על המשתמשים, רק כאלו שרלוונטיים למקצה ההרצה, כלומר משתמשים מחוברים, או כאלו שהם בעלי חנות, מנהלי חנות, או מנהלי מערכת.
2. עבור משתמשים נטענים רק השדות של שם משתמש וסיסמה, ודגל האם המשתמש מחובר. כלומר לא נטען אובייקטים כבדים כמו עגלת קניות, אלא רק בעת ההתחברות והצפייה בעגלה, ולא היסטוריית רכישות אלא רק בעת בקשה של צפייה בהן.
3. חנויות נטענות ללא היסטוריית הרכישות שלהן וללא מדיניות הרכישה וההנחות אלא רק בעת בקשת צפייה בהן בהיסטוריה, או ברכישה.
4. כאשר משתמש מבצע Log out הוא ימחק מה identity map ב mapper שוב כדי למנוע עומס על הזיכרון הראשי ולתחזק רק את המשתמשים הרלוונטיים למקצה ההרצה הנוכחי.

* תמונה שמכילה טקסט

  התיאור נוצר באופן אוטומטישינוי ב API כך שיתמוך ב 500 בקשות במקביל.

בפרויקט API .NET Core מספר הבקשות שיכולות להיכנס בו זמנית הוא 10 באופן דיפולטיבי.   
על מנת שנוכל לתמוך במקביליות לפי המדדים שהגדנו, היה עלינו לפתוח את ה API לכמות גדולה יותר של בקשות במקביל על מנת לחסוך תורי המתנה של thread- ים למענה לבקשות ולחסוך בזמנים.

* השתמשנו בכלי Jmeter שהוא כלי לבדיקות קיבול ועומס, המאפשר הגדרה של מספר התרדים ובכמה שניות הם ירוצו ויבצעו את הבקשות.

יתרון נוסף של Jmeter הוא שיכולנו בעזרתו להגדיר ולהגריל שמות רנדומליים של אימיילים, שמות חנויות וכו, ומאפשר ליצור כמות אובייקטים רבה בקלות ובעזרת אוטומציה.

* על מנת להקל על יצירת והרצת התרחישים השתמשנו ב Blaze meter שהוא תוסף chrome שמאפשר לשמור הקלטת פעולת הרצה (למשל פתיחת חנות) בפורמט של jmeter.

**ניתוח תרחישי בדיקה –**

**בדיקות עומס stress -**

ראשית ניצור 100 משתמשים בשמות ייחודיים

**תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי**

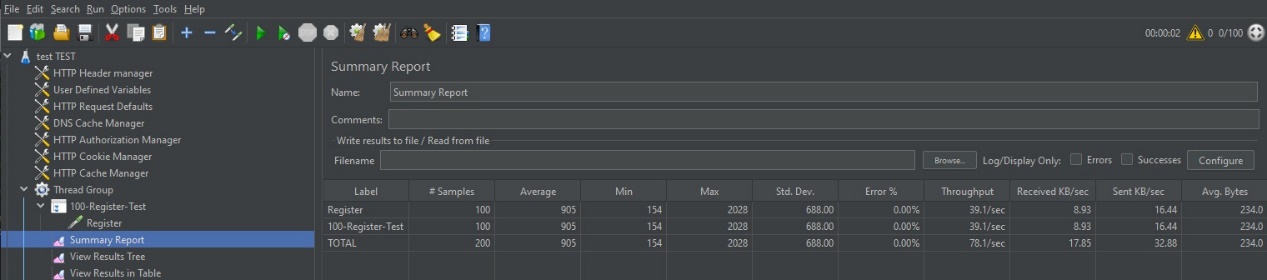
כעת נגדיר מקביליות בעזרת Jmeter שיריץ 100 תרדים בשנייה אחת בלבד.

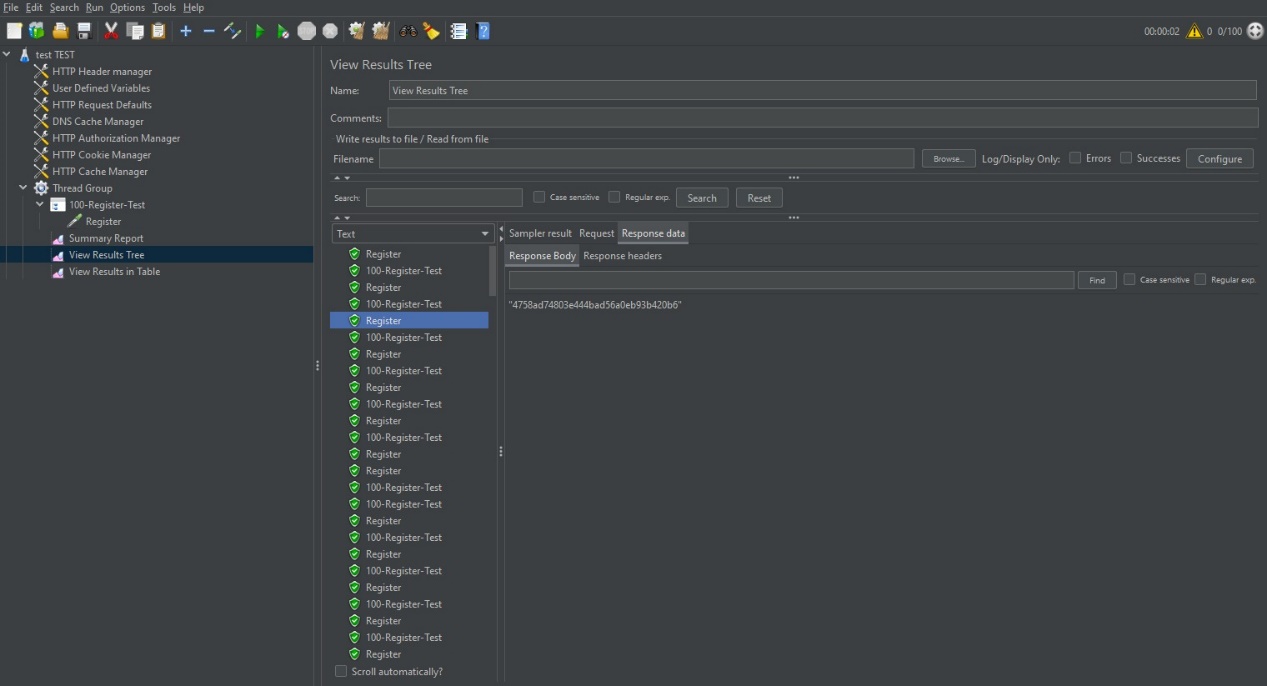
תמונה שמכילה טקסט

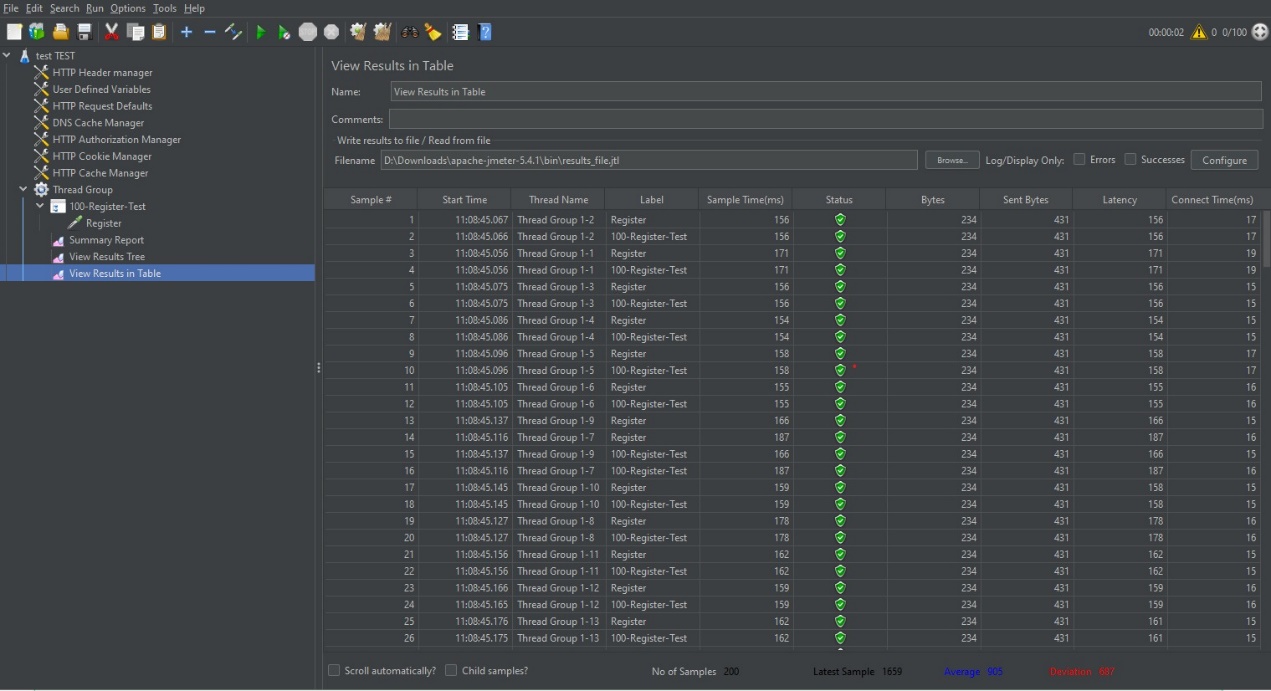
התיאור נוצר באופן אוטומטי

כעת נמדוד את הביצועים, ראשית נבדוק שכל הבקשות התקבלו וקיבלו מענה מהשרת, כלומר שלא היה שגיאות בהתחברות לשרת או בקשות שנכשלו.

שנית נמדוד זמנים, נשים לב שאנחנו מעוניינים למדוד את הזמן הממוצע שבקשה מהשרת קיבלה מענה, ולא את משך הרצת הטסט שהוא חסר משמעות בהקשר הזה.







ניתן לראות שכל הבקשות עברו ושהממוצע למענה של בקשה הוא 0.9 שניות, כאשר מחצית מהבקשות עברו ב 0.2 שניות.

כעת נבחן דבר נוסף, במידה והיינו מריצים את 100 התרדים הנ"ל ב 5 שניות (עדיין מסמלץ מקביליות אמיתית), אנחנו רואים שיפור משמעותי בזמנים.

נזכור שאנחנו לא מסתכלים על זמן הרצת הטסט אלא על זמן המענה לכל בקשה.

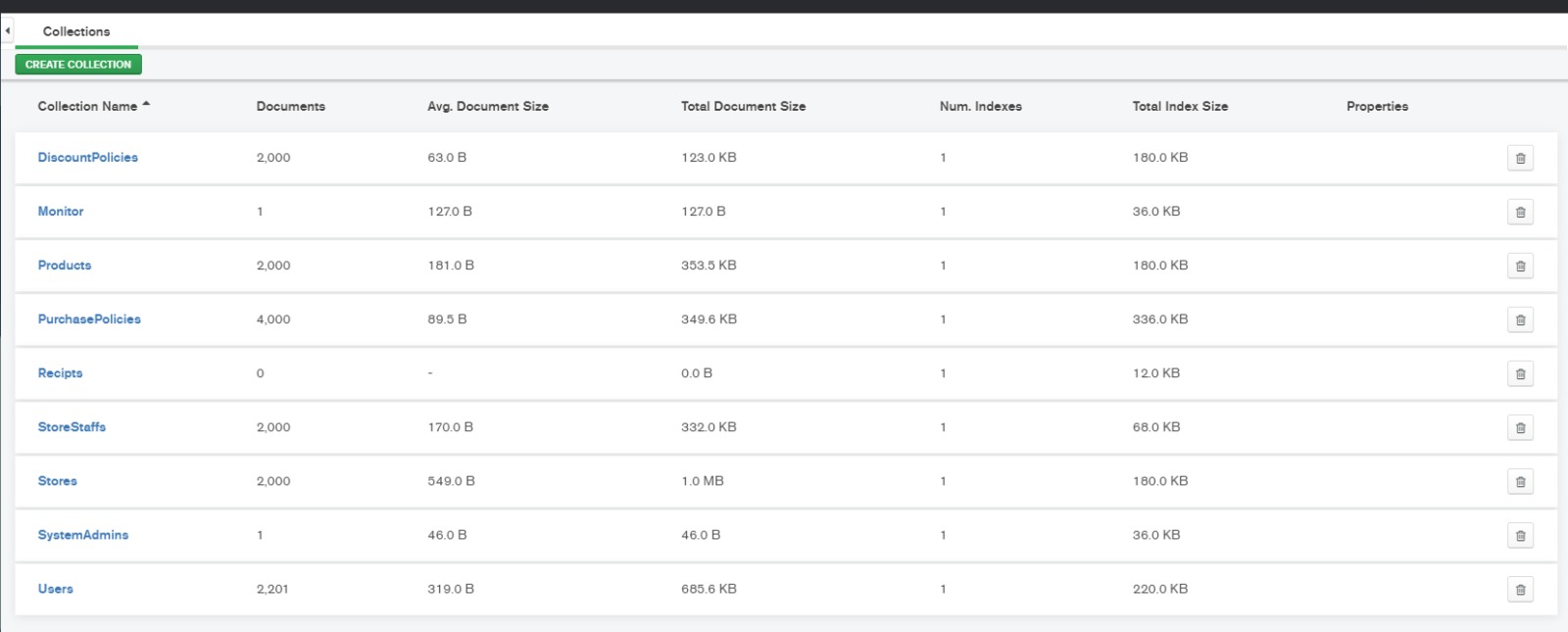
תמונה שמכילה טקסט, צילום מסך, אלקטרוניקה, מחשב

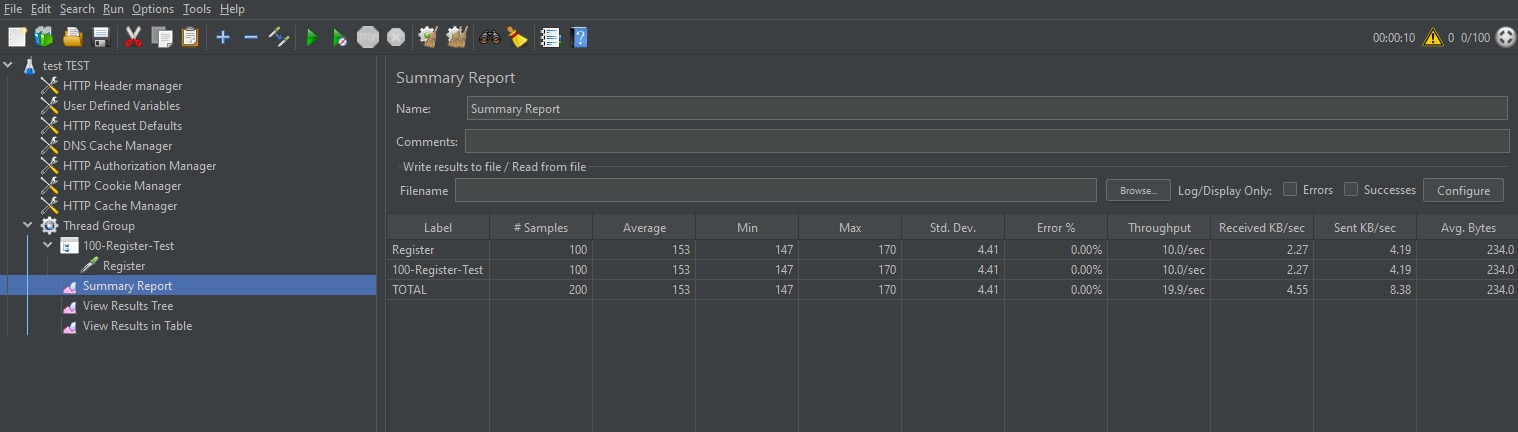
התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

ניתן לראות שבפריסת הבקשות על פרק זמן ארוך יותר הממוצע למענה לבקשה יהיה 0.152 שניות !!

כעת ננסה לראות האם עומס בבסיס הנתונים משפיע על זמן שליחת וקבלת המענה על הבקשות שלנו. לצורך כך הכנסנו מידע רב מאוד של מידע והרצנו פעם נוספת את הטסט.

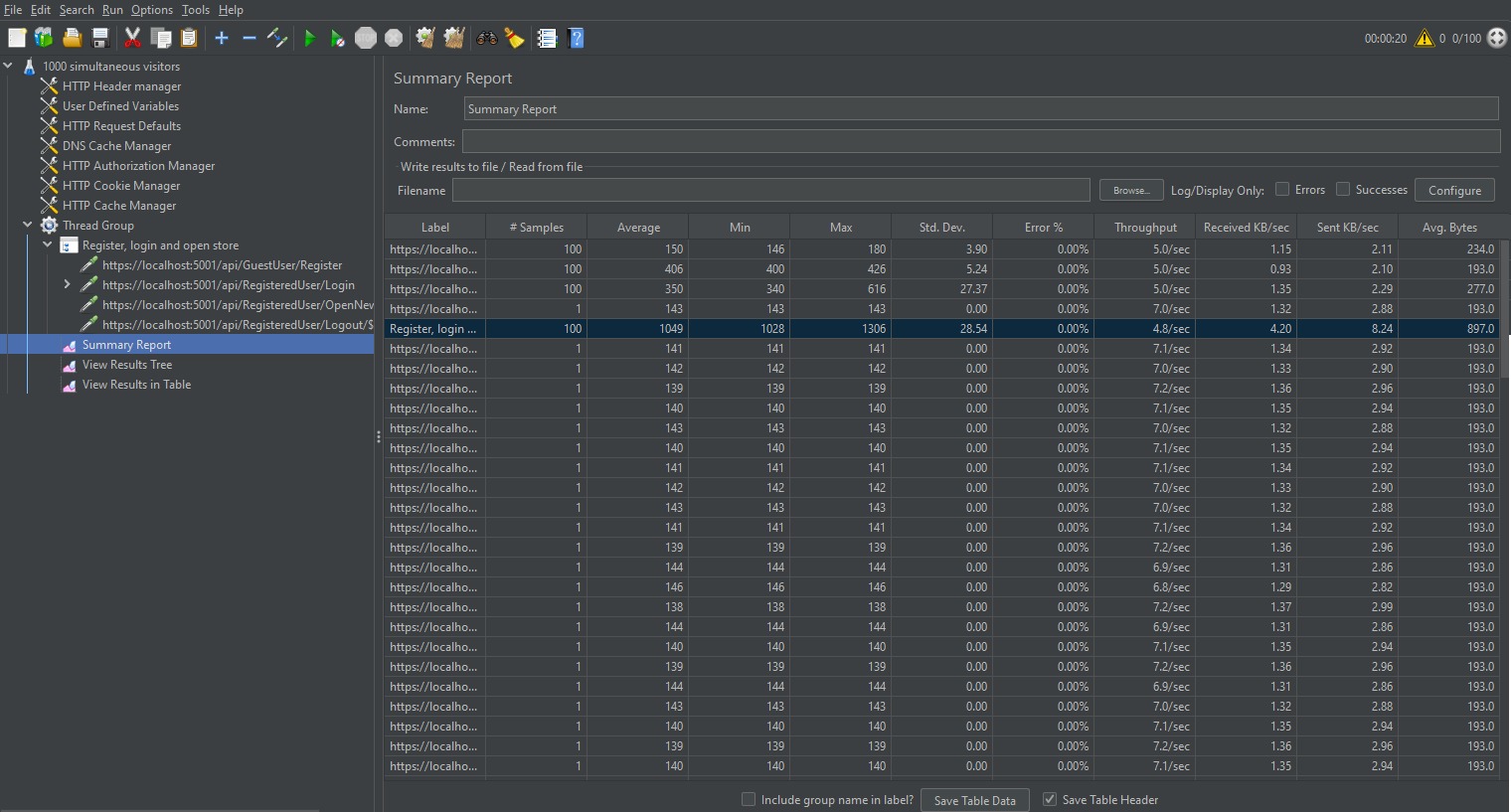




נראה כי אין השפעה על זמן קבלת הבקשה הממוצע שנשאר 0.153 שניות.

כעת נבחן פונקציונליות נוספת של המערכת בהוספת חנות חדשה, נריץ 100 תרדים שונים לביצוע המשימה הזאת.

נזכור- על מנת לפתוח חנות חדשה במערכת יש צורך קודם כל ב – 100 בקשות הרשמה, 100 בקשות התחברות, 100 בקשות פתיחת חנות, 100 בקשות log out בתרחיש שלנו.

נראה שכל הבקשות עברו.

**בדיקות קיבול Load –**

בבדיקות אלו בעצם נרצה לוודא שהמערכת נטענת ומצליחה לתפקד תחת מספר רב מאוד של משתמשים בבסיס הנתונים.

נציג את העומס שיצרנו לצורך הבדיקות, ע"י שמירה של מאות אובייקטים, משתמשים חנויות ומוצרים.

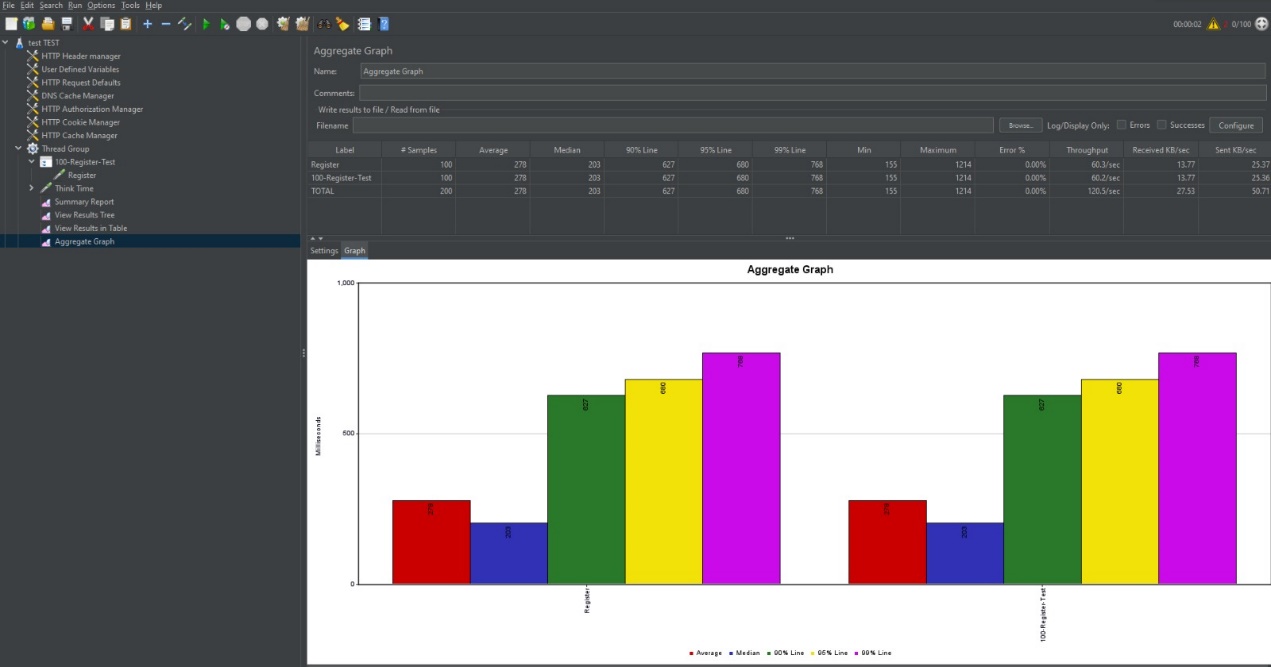
תמונה שמכילה טקסט, מקורה, צילום מסך

התיאור נוצר באופן אוטומטיראשית נוודא שהמערכת נטענת, כאמור הטעינה היא עצלה ולכן ישנו שיפור משמעותי של זמני הטעינה.

לאחר מכן ביצענו פעולות בסיסיות במיוחד על המערכת לוודא את הטעינה של הכמות המאסיבית של אובייקטים.

לבסוף, הרצנו על המערכת הנ"ל תרחישי עומס נוספים, לראות שאינה קורסת ומצליחה לשרת את כל המשתמשים והבקשות בזמנים סבירים.

נמחיש בגרף, הרצה של 100 תרדים במהלך 2 שניות, כאשר בסיס הנתונים עמוס במיוחד.   
נציין שכדי לייצר את הגרף עשינו עיכוב (delay) של 2 שניות על מנת לעבד את הדאטה, מה שגם הוביל לעיכוב הריצה של התרדים.



נראה כי העמודה הצהובה המתארת 95% מבקשות ההתחברות לא עולה על שנייה אחת כנדרש.