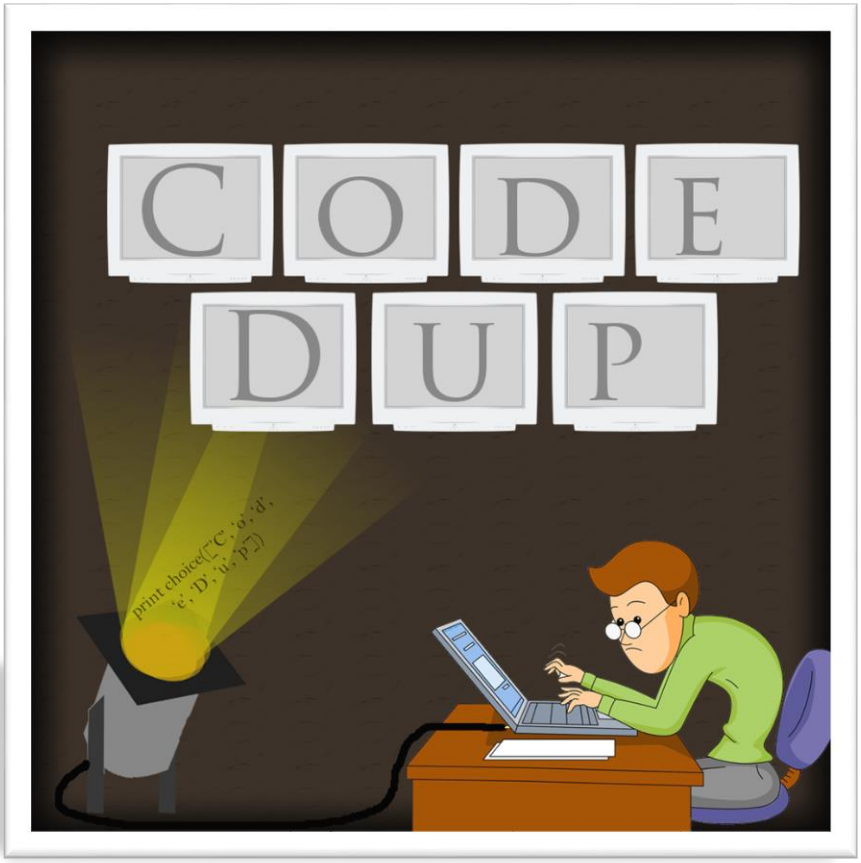
פרויקט בסייבר, 5 יח"ל:

CodeDup



מגיש: מאור רביח

ת"ז: 318879608

בי"ס: אורט בנימינה

מורים: מרי גבע, ערן פרי, גל שקד

תאריך: מאי 2017

תוכן העניינים

פרק א' – פרק הייזום

תיאור ראשוני של המערכת

תיאור כולל ורציונלי על הפרויקט

המערכת מתוחמת בתוך גבולות מעבדת מחשבים (על עף שהיא צפויה לתפקד כראוי גם מחוץ לרשת מקומית) והיא מורכבת ממשתמשים, מוציאים לפועל ומפעיל.

כל משתמש יכול לשלוח למערכת בקשה להרצת קובץ מסוים בעוד הבקשה עוברת דרך מחשב המפעיל. מחשב המפעיל סורק את המערכת וקובע מתוך שיקולים שונים האם הבקשה יכולה בכלל להתממש או שמה היא חורגת מהגבולות שהמערכת נתונה אליהם ברגע הנוכחי. במידה והבקשה נמצאה עומדת בסטנדרטים, היא נשלחת אל המוציאים לפועל שנבחרו על ידי מחשב המפעיל בעוד שהם, מוציאים אותה לפועל. המשתמש, באופן ישיר, רואה את התקדמות הבקשה שלו אצל כל המוציאים לפועל שלה, כל אחד בנפרד.

כמו כן, למפעיל יש שליטה מסוימת על הנעשה במערכת, הוא יכול לנתק משתמשים ומוציאים לפועל, להפסיק הרצתן של בקשות ולהגביל את כמות כל אחד מהם במערכת.

מבוא ורקע כללי לנושא העבודה

הנושא הכללי בא לפתור בעיה שכיחה בכיתה, שבה משתמש נאלץ להסתובב בכיתה, לחפש מחשבים פנויים ולהפריע למשתמשים אחרים על מנת להריץ את הקובץ שלו.

מה המוצר המוגמר אמור לבצע

המוצר המוגמר אמור לספק למשתמש ממשק נוח לשם הרצת קבצים על מחשבים שונים ברשת.

הגדרת הלקוח

למי מיועדת המערכת ומי הולך להשתמש בה?

המערכת מיועדת לכל מפתח שמתעסק עם קובץ שאותו הוא צריך להריץ במספר מחשבים במקביל.

הגדרת יעדים/מטרות

מה המטרות המרכזיות של המערכת המוצעת?

- להריץ את הקובץ על מחשב אחר בכיתה בצורה חלקה כמצופה מהמשתמש.
- לאפשר למשתמש לעקוב אחר הקובץ הרץ.
- לאפשר למפעיל לשלוט על המצב של המערכת בכל רגע נתון.

בעיות, תועלות וחסכונות

מה הבעיה ומה אנחנו מנסים להשיג?

הבעיות מפורטות להלן:

- הרצת קטע הקוד על מחשבים אחרים בכיתה מצריך בחירת מחשבים- ייתכן שאין מספיק מחשבים פנויים ומכאן שצריך להפריע למחשב תפוס.
- קיימים מחשבים עם מעט מידי משאבים זמינים, צריך לבצע בחירה מושכלת.
- לאחר הרצת קטעי הקוד יש צורך ניטורי, לפעמים אפילו באופן מקבילי.

כוונתנו היא שהמערכת תדע לטפל בכל הבעיות הנ"ל באופן היעיל ביותר.

מה התועלות שסביר לצפות מהמערכת?

סביר לצפות מהמערכת שתוכל לטפל בבקשתו של המשתמש, שהיא הרצת קובץ על מספר מחשבים בכיתה. כמו כן, היא תספק לו ממשק שדרכו יוכל לעקוב כיצד הקובץ רץ באופן ישיר ותהיה נוחה לשימוש.

אילו שירותים המערכת תיתן?

תספק לכל משתמש ממשק גרפי בהתאם לצרכיו ולמחשב הראשי תספק ממשק בקרתי שבאמצעותו יוכל להגביל ולווסת את התנהלות המשתמשים האחרים בכיתה ולהציג נתונים כדוגמת פלט התכנית ומצב צריכת המשאבים.

השוואת העבודה עם פתרונות ויישומים קיימים

לא נמצאו יישומים או פתרונות קיימים נכון לזמן זה.

האם צפויים קשיים או מגבלות בהגדרת המערכת?

האם מדובר בטכנולוגיה חדשה ובלתי מוכרת?

לא נדרש שימוש בטכנולוגיה כזו.

האם קיימים סייגים בהגדרות המערכת?

- מספר בקשות מרובה ועומס רב ידרשו את ניצול המשאבים היעיל ביותר.
- במקרה של עומס כבד, עלול להיווצר מצב שבו קריטי למשתמש שהקובץ שלו המיועד להרצה, ירוץ באופן חלק על ידי המוציאים לפועל לשם סנכרון עם המריצים האחרים, בדיקת היעילות של התכנית המורצת ועוד. כאמור במקרה של עומס לא בטוח שהמערכת תוכל לעמוד בדרישות מסוג זה.

תיחום הפרויקט

התחומים בהם הפרויקט עוסק

רשתות- הממשק של המשתמשים בכיתה מתקשר עם מחשב המפעיל, בעוד שזה מתקשר עם כל המחוברים למערכת. התקשורת נעשית באמצעות הרשת.

מערכות הפעלה- על מנת להריץ את הקובץ על מחשב אחר יש להתעסק עם מערכת ההפעלה.

המודולים והנושאים בהם המערכת מטפלת

- **תקשורת** – סביבה הבנויה משרת ולקוחות.
- **הרצת קובץ אוטומטית** – הרצת קובץ במחשב ללא משתמש שיעשה זאת.
- **שיתוף תמונת מצב** – הצגה למשתמש תמונות מצב לגבי הקובץ הרץ.
- **אבטחה** – אבטחת התקשורת בין מחשב המשתמש למחשבים האחרים.
- **ממשק משתמש גרפי** – ממשק גרפי גם לשרת וגם ללקוחות.
- המערכת לא תספק שיתוף מסך.
- המערכת לא תריץ את הקובץ על מחשב זר ללא אישור.

פרק ב'- פרק האפיון

פרוט המערכת

השרת הוא שרת מרובה לקוחות, בהתחברות הראשונה של כל מחשב הוא נרשם בשרת. השרת מנהל בסיס נתונים ובו מידע רלוונטי על כל מחשב ומחשב- האם הוא מחשב של מוציא לפועל או משתמש, מה כתובת האיי-פי שלו, ממתי הוא מחובר למערכת, האם הוא פעיל ועוד.

בכל רגע נתון יכול אחד המשתמשים לשלוח לשרת בקשה. הבקשה נועדה להסביר לשרת מהם בדיוק הדרישות לשם הרצת הקובץ, כמה מוציאים לפועל הוא דורש, הקובץ להרצה עצמו, קבצים נלווים אם יש כאלה ועוד.

השרת, בליווי בסיס הנתונים צריך לבדוק את תקינות הבקשה שהתקבלה ולקבוע האם היא עומדת בגבולות הנוכחיים של המערכת. ברגע שהוא מסיים את הבדיקה וקובע מי יהיו המוציאים לפועל של הבקשה שהתקבלה, הוא שולח את המידע הנחוץ על הבקשה למוציאים לפועל ואלה מוציאים אותה לפועל, כל אחד על המחשב שלו. מרגע זה השרת לא אחראי על מה שמתנהל בקשר לבקשה, אלא רק מעביר את הנתונים אודותיה מצדם של המוציאים לפועל אל המשתמש שהבקשה בבעלותו. למפעיל היכולת לעצור את פעילות הבקשה בכל רגע שהוא.

הממשק הגרפי של המשתמש יציג לו בכל רגע נתון תמונת מצב של כל המחשבים שמריצים את הקובץ שלו. תמונת המצב כוללת את פלט הקובץ, שגיאות אם יש כאלה וכל מידע שימושי אחר.

מה היכולות שהיא תעניק למשתמש

משתמש הלקוח

I. שליחת בקשה לשרת – המשתמש יצטרך לבנות את הבקשה שלו לשרת.
   i. לציין מהו הקובץ שברצונו להריץ.
   ii. לקבוע לכמה מוציאים לפועל הוא זקוק.
   iii. לצרף קבצים נלווים אם ישנם.

II. מעקב אחר הרצת הקובץ – המשתמש יוכל להציג את כל הנתונים ההכרחיים לשם מעקב ועדכון אודות השלכות הקובץ- כל מחשב והנתונים הרלוונטיים לו.

המפעיל בשרת

III. הגבלה – משתמש השרת יוכל לנהל מערך הגדרות והגבלות שהמשתמשים והמערכת יהיו כפופים אליהם.

IV. מעקב ישיר – משתמש השרת יוכל להציג את ההתנהלות במעבדה באופן ישיר, מה הקבצים שמריץ כל מוציא לפועל ונתונים נוספים הנשמרים בבסיס הנתונים.

פירוט הבדיקות ('קופסא שחורה')

מספר	שם הבדיקה	מה אמורה לבדוק	איך מתכננים לבדוק
1	קומבינציית הגדרות וקלטים בלקוח	האם המערכת מטפלת כמצופה תוך כדי שילוב הגדרות שונות בלקוח.	שליחת בקשות לשרת עם כל הקומבינציות האפשריות. הציפיה מהשרת שיצליח להצליב ביניהן וימשש את כולן.
2	קומבינציית הגדרות וקלטים בשרת	האם המערכת מטפלת כמצופה תוך כדי שילוב הגדרות שונות בשרת.	לתת למערכת להתנהל תוך ניסוי כל הקומבינציות האפשריות בהגדרות של השרת. הציפיה מהמערכת שתתנהל לפי סדר עדיפויות הגיוני בעת ניגוד דרישות בין בקשת הלקוח להגבלות השרת.

מספר	שם הבדיקה	מה אמורה לבדוק	איך מתכננים לבדוק
3	העמסה	האם המערכת מצליחה לעמוד בעומסים.	להפעיל מספר בקשות במקביל, לנצל את מירב המשאבים בכיתה. הציפיה מהמערכת שתתנהל ביעילות ותישאר יציבה על עף העומס.
4	אבטחה	לבדוק האם המערכת מאובטחת ברמה טובה.	לזייף לקוחות (זיהוי מחשבים שלא הוכרו על ידי השרת ומנסים לנצל את משאבי הכיתה – הצלבה עם מספר המחשבים בכיתה), להשתמש בהגדרות זדוניות (sql injection בשדות בעלי תקשורת ישירה עם בסיס הנתונים), הצפת בקשות לשרת (Dos-Attack) – מצופה מהשרת לזהות מקרה כזה והסנפה (בדיקה האם התעבורה מוצפנת ולא ברורה לman in the middle).
5	טעויות מכוונות	האם המערכת מצליחה לעמוד בהתנהלות בלתי מצופה מצד הלקוח.	להשתמש במערכת בניגוד להוראות, להכניס מחרוזת איפה שצריך מספר, לצרף תמונה איפה שקובץ הרצה מבוקש וכו'. הציפיה

מספר	שם הבדיקה	מה אמורה לבדוק	איך מתכננים לבדוק
			מהמערכת שתעיר על כך ושהדבר לא יערער את יציבותה.
6	הרצת הקובץ כמצופה	האם המערכת מצליחה להתמודד עם הרצת קבצים המבצעים פעולות שונות.	כתיבת סקריפטים המבצעים פעולות שונות ובדיקה האם המערכת מריצה אותם כנדרש. הציפיה מהמערכת שתצליח להריץ אותם כפי שהקובץ היה רץ במחשב הלקוח.
7	תמונת מצב נכונה	האם תמונת המצב שמתקבלת תואמת את הפלט שמתקבל.	הרצת קבצים שונים ובדיקה האם תמונה המצב אכן עדכנית ותואמת את הנתונים האחרים. הציפיה מהמערכת שתצליח להעביר את תמונת המצב בקצב סביר בהצלחה.
8	ממשק לקוח תקין	האם ממשק הלקוח עובד כראוי ולא מאפשר לו לעשות דברים שהמערכת לא התחייבה לספק.	לחיצה על כפתורים ללא הכר וניסיונות לנצל את הממשק לדברים שהוא לא מיועד לעשות. הבדיקה תכלול גם שימוש בכל הכלים שהוא מציע ובדיקה האם הם עובדים כראוי. הציפיה מהמערכת

מספר	שם הבדיקה	מה אמורה לבדוק	איך מתכננים לבדוק
			שתצליח להפעיל ממשק משתמש מוגבל ותקין.

תכנון לוח זמנים לפרויקט

פעילות	זמן התחלה מתוכנן	זמן סיום מתוכנן	זמן התחלה בפועל	זמן סיום בפועל	הערות
פרק הייזום	1/12/16	10/12/16	4/12	7/12	
פרק האפיון	10/12/16	17/12/16	7/12/16	18/12	
פרק הניתוח	17/12/16	2/1/17	18/12/16	31/1/17	
פרק העיצוב	2/1/17	1/2/17	31/1/17	13/2/17	
תשתית לשרת + לקוח.	1/2/17	3/2/17	14/2/17	20/2/17	תשתית מוציא לפועל התווספה בהמשך עם שינוי מבנה המערכת שהתקבל מאוחר יותר.
הגדרת מחלקות ראשיות- שרת + לקוח	3/2/17	4/2/17	17/2/17	19/2/17	מכאן ואילך, שרת לקוח שונה למוציא לפועל, הנפרד מהלקוח עצמו.

פעילות	זמן התחלה מתוכנן	זמן סיום מתוכנן	זמן התחלה בפועל	זמן סיום בפועל	הערות
בניית שרת לכל לקוח	4/2/17	6/2/17	20/2/17		בוטל.
תשתית תקשורת בין לקוח + שרת לקוח + שרת ראשי	6/2/17	8/2/17	20/2/17	22/2/17	
שליחת קובץ מהלקוח לשרת, משם לשרתי הלקוחות והרצתו	8/2/17	12/2/17	22/2/17	28/2/17	
תשתית ממשק גרפי לקוח	12/2/17	14/7/17	29/2/17	12/1/17	
שאיבת נתונים מהלקוחות ובניית בסיס נתונים	14/2/17	18/2/17	12/1/17	14/1/17	
מימוש האלגוריתם לטיפול בבקשה	18/2/17	22/2/17	14/1/17	17/1/17	
טיפול בממשק הגרפי של	22/2/17	25/2/17	17/1/17	25/1/17	

פעילות	זמן התחלה מתוכנן	זמן סיום מתוכנן	זמן התחלה בפועל	זמן סיום בפועל	הערות
הלקוח והתקשורת עם שרתי הלקוחות האחרים					
השלמת חותר, שלד ראשוני	-	-	25/1/17	3/3/17	לא נלקח בחשבון.
הוספת שיפורים ותכונות מיוחדות	-	-	4/3/17	26/4/17	לא נלקח בחשבון.
שפצורים אחרונים	25/2/17	1/3/17			
סיום פיתוח	1/2/17	1/3/17			
מסמך בדיקות סגור + סיום תיקון שגיאות	1/3/17	30/3/17	28/4/17		
מדריך למשתמש	30/3/17	8/4/17	28/4/17		
הגשת תיק פרויקט שלם והפרויקט עצמו	8/4/17	26/4/17	29/4/17		

ניהול סיכונים בפרויקט

הסיכון	פירוט הסיכון	רמת הסיכון	תיאור דרכים להתמודדות עם הסיכון	מה בוצע בפועל	תאריך
קבצים זדוניים	בדיקה שהקובץ המיועד לריצה לא מבצע פעולות זדוניות.	קשה	-פתרונות יצירתיים עקיפים כמו מתן דירוג לכל לקוח. -פיתוח רק גרסה בסיסית לתכונה.	יש אפשרות להריץ את המוציאים לפועל מתוך מכונה וירטואלית.	18/4/17
עומס	המערכת יכולה להיקלע למצב של עומס כבד. הדבר יכול לבוא לידי ביטוי עם פלט מאסיבי וריבוי בקשות.	בינוני	-הגדלת קצב התמונות לשנייה קצב עדכון המשאבים.	פיתוח המערכת כיציבה עם תשתיות שיכולות לעמוד בעומס גבוה תוך שמירה על יציבות.	לאורך כל הפיתוח.
חווית ממשק נוחה	יהיה על ממשק הלקוח להיות נוח לשימוש. הדבר עלול להיות מעט	קלה	-תכנון הממשק עד הפרט האחרון לפני יצירתו. -למידה מעמיקה יותר לעבודה עם	הושקע זמן רב בפיתוח ממשק נוח למשתמש. פיתוח החלון למעקב אחר המוציאים	20/3/17

	מסובך למפתח שמתעסק עם המודול הגרפי הנבחר בפעם הראשונה בסדר גודל הנתון.		המודול של הממשק.	לפועל לווה בחקר ותכנון משמעותיים.	

פרק ג'- פרק הניתוח

פרוט היכולות של המערכת

התחברות לשרת

כל מחשב של משתמש בכיתה (שרוצה להשתתף במערכת) צריך להירשם למערכת.

מהות היכולת היא להעניק זיהוי לכל מחשב בכיתה. בעזרת הזיהוי יהיה ניתן לאגור עליו מידע במסד הנתונים ולקשר בינו לבין המשתמש שיריץ עליו את הלקוח. לקוח יוכל להיעזר במסד הנתונים של השרת על מנת לדעת את כמות המשאבים שהקובץ שלו צורך בכל מחשב ונתונים רלוונטיים נוספים.

המשתמש יוכל להיעזר בו לשם מעקב אחר מחשב מסוים. לדוגמה משתמש בוחר להריץ קובץ על מחשבים: 2,4,7. הוא מבחין במשהו מוזר שקורה במחשב 4. לכן הוא מעדכן את הקוד שלו ומעוניין לבדוק האם הקוד המעודכן פותר את הבעיה שקרתה במחשב 4 בהרצה הקודמת. בפועל, בחלון שבו המשתמש רואה את המעקב אחר הקובץ שלו בכל אחד מהמחשבים המריצים, כתוב ליד כל פלט את מספר המחשב שאחראי עליו. משתמש יכול לערוך בקשה כך שיבקש מהשרת להריץ את הקובץ על מחשב מסוים לפי מספר הזיהוי שלו.

אוסף התהליכים שהיכולת צריכה לבצע:

I. עם הדלקת המחשב יהיה עליו להתחבר לשרת, הדבר יעשה על ידי אחד המשתמשים בכיתה או על ידי המשתמש של המחשב הנוכחי.
II. כל מחשב המיועד להיות חלק מהמערכת ברגע ההתחברות לשרת מקבל סימן זיהוי באופן אוטומטי מהשרת (אמנם הip של כל מחשב במערכת ייחודי, אך יותר פרקטי ונוח למשתמש לזכור מספר על פני כתובת ip).
III. השרת מעדכן את בסיס הנתונים שלו, מצרף את המחשב כחבר חדש במערכת ומתחיל לעדכן את הנתונים הרלוונטיים לו.

רשימת האובייקטים בהם היכולת עושה שימוש:

- תקשורת – תהליך ההתחברות של המחשב לשרת עושה שימוש בתקשורת. המחשב שולח לשרת שהוא מעוניין להתחבר למערכת, השרת מבצע את תהליך ההתחברות ומשיב למחשב שהוא התחבר למערכת בהצלחה.
- מסד הנתונים של המערכת – היכולת תעשה שימוש במסד הנתונים של השרת- היא תעדכן בכל המקומות הנחוצים את עצם קיומו של המחשב החדש במערכת.
- אובייקטים נוספים: לקוח, שרת.

<u>ליצור קובץ ולכוון אליו את פלט התכנית</u>

מהות היכולת היא לשמור את פלט הקובץ הרץ.

<u>אוסף התהליכים שהיכולת צריכה לבצע</u>:

I. ליצור קובץ תיעוד פלט שבו ישמר הפלט של התכנית.
II. לשנות את הstdout של מערכת ההפעלה כך יהיה מכוון אל הקובץ.

<u>רשימת האובייקטים בהם היכולת עושה שימוש</u>:

קובץ רץ, קובץ תיעוד פלט, מערכת ההפעלה.

<u>ליצור קובץ ולכוון אליו את הודעות התעופה של התכנית</u>

מהות היכולת היא לשמור את הודעות החריגה בזמן שהתכנית קורסת.

<u>אוסף התהליכים שהיכולת צריכה לבצע</u>:

I. ליצור קובץ תיעוד חריגות שיכיל את הודעת התעופה של התכנית.
II. לשנות את הstderr של מערכת ההפעלה כך יהיה מכוון אל הקובץ.

<u>רשימת האובייקטים בהם היכולת עושה שימוש</u>:

קובץ רץ, קובץ תיעוד חריגות, מערכת ההפעלה.

מחיקת קלטים בצד הלקוח

היכולת למעשה מתחילה לפעול כשהיא מקבלת התראה כי קיים קלט לא תקין מהיכולת שבודקת קלטים בלקוח.

מהות היכולת היא למחוק את הקלט הלא תקין מממשק הלקוח.

אוסף התהליכים שהיכולת צריכה לבצע:

I. להבין שקיים קלט לא תקין.
II. למחוק את הקלט ממשקו הגרפי של הלקוח.

רשימת האובייקטים בהם היכולת עושה שימוש:

התראה, קלט לא תקין, מחיקה, ממשק הלקוח.

ההתראה על קלט לא תקין בצד הלקוח

היכולת למעשה מתחילה לפעול כשהיא מקבלת התראה כי קיים קלט לא תקין מהיכולת שבודקת קלטים בלקוח.

מהות היכולת היא להודיע ללקוח כשהוא מכניס קלט לא תקין ומה הייתה הבעיה בקלט שהכניס.

אוסף תהליכים שהיכולת צריכה לבצע:

I. להבין שקיים קלט לא תקין.
II. להדפיס הודעה ללקוח כי הקלט שהכניס לא תקין ולהסביר מה הבעיה בקלט זה.

רשימת האובייקטים בהם היכולת עושה שימוש:

התראה, קלט לא תקין, הדפסה.

בדיקת קלטים (לקוח)

מהות היכולת לתפוס את הקלטים שהמשתמש הכניס וחרג מהטווח שהומלץ לו.

אוסף התהליכים שהיכולת צריכה לבצע:

I. היכולת עומדת מאחורי כל קלט שהשרת עדכן טווח מסוים של ערכים אפשריים.
II. היכולת מזהה את הקלט ובודקת האם הוא עומד בתנאים שהוצבו למשתמש.
III. במידה והיכולת מזהה אי תקינות בקלט היא מודיעה על כך ליכולת שמוחקת קלטים וליכולת שמודיעה למשתמש כי הקלט שהכניס לא תקין.

אוסף האובייקטים שהיכולת עושה בהם שימוש:

קלטים שגויים, מפתח, תנאים, משתמש, טווח אופציות לקלט.

התעדכנות מהשרת ועדכון קלטים אפשריים

היכולת מצפה כל הזמן לעדכון מהשרת בדבר שינוי קלט אפשרי.

מהות היכולת היא להשאיר את המשתמש מעודכן כל הזמן למצב המערכת ובכך למנוע ממנו מלכתחילה הכנסה של קלט שהמערכת לא יכולה לעמוד בו. לדוגמא אם השרת יודע שרק שישה מחשבים מחוברים למערכת הוא יודיע זאת ליכולת בכדי שתעדכן את המשתמש שאין טעם שיבקש יותר משישה מחשבים כי השרת לא יוכל להוציא את הבקשה לפועל. היכולת מקלה משמעותית על השרת.

אוסף התהליכים שהיכולת צריכה לבצע:

I. להיוודע מהשרת שקלט אפשרי השתנה (יש לשים לב כי היכולת עצמה לא מאזינה לשרת אלא מקבלת את ההודעה מהשרת בעזרת היכולת: "ניווט החבילות בהתאם לפרוטוקול התקשורת").
II. להודיע ליכולת: "ממשק המשתמש" שעליה לשנות את הקלט האפשרי שהיא מציגה למשתמש.

רשימת האובייקטים בהם היכולת עושה שימוש:

קלט אפשרי.

עדכון המשתמשים בדבר הקלטים האפשריים

היכולת פועלת כל הזמן.

מהות היכולת היא להודיע למשתמשים במערכת על פי מצב המערכת מהם הקלטים האפשריים שהם יכולים להכניס בבקשות.

אוסף התהליכים שהיכולת צריכה לבצע:

I. בדיקה רציפה של מסד הנתונים וחיפוש שינויים שיכולים להשפיע על השרת להחליט שהבקשה שהתקבלה לא יכולה לצאת לפועל.
II. להכין את הקלט האפשרי החדש שנמצא בחבילה שתתאים לתנאי הפרוטוקול.
III. לשלוח את הקלט האפשרי לכל הלקוחות המחוברים למערכת.

רשימת האובייקטים בהם היכולת עושה שימוש:

מסד הנתונים, בקשה, קלט אפשרי.

שליחת בקשה לשרת

מהות היכולת לספק למשתמש אפשרות לשלוח לשרת דרישה המפרטת את ההגדרות שעל פיהם הוא יצטרך להתנהל, את הקבצים שאיתם הוא יעשה שימוש ופירוט המשאבים להם הוא זקוק.

אוסף תהליכים שהיכולת צריכה לבצע:

I. סיפוק ממשק למשתמש שבו יוכל להכניס את כל מה שהבקשה תכיל, קובץ הקלט, קובץ ההרצה, הגדרות ונתונים רלוונטיים נוספים (היכולת הנוכחית לא אחראית לגרפיקה של הממשק).
II. מתוך קלט המשתמש ליצור בקשה על פי הפרוטוקול של המערכת שהשרת יוכל להבין.
III. שליחת הבקשה לשרת.

רשימת האובייקטים בהם היכולת עושה שימוש:

קובץ הרצה, קובץ קלטים, פרוטוקול תקשורת, בקשה.

פענוח מסרים

היכולת פועלת בכל המחשבים המחוברים למערכת ומשתמשים בפרוטוקול התקשורת.

מהות היכולת היא לפענח מסרים (בקשה, הוראה או כל הודעה שהיא) שהתקבלו וצפויים להגיע כמוצפנים. כלומר, אם נקבע מראש שחלק בחבילה מסוימת שהתקבלה מכיל תוכן מוצפן, היכולת תדע שעליה לפענח אותו. כמו כן היכולת יודעת באיזה שיטת פיענוח עליה להשתמש. למעשה, הדבר הראשון שכל ישות במערכת עושה לאחר שקיבלה חבילה באמצעות פרוטוקול התקשורת הוא לפענח אותה.

אוסף התהליכים שהיכולת צריכה לבצע:

I. לקבל את החבילה שהתקבלה מהיכולת שקלטה אותה באמצעות פרוטוקול התקשורת.
II. לזהות את החבילה שהתקבלה (האם זו בקשה, הוראה או כל דבר אחר) לפי הכותרות שלה.
III. לפי סוג החבילה לפענח את החלקים שאמורים להיות מוצפנים באמצעות דרך הפיענוח הרלוונטית להם.

רשימת האובייקטים שהיכולת עושה בהם שימוש:

פרוטוקול תקשורת, מסר, פיענוח, מוצפן, חבילה, ישות.

הצפנת מסרים

היכולת פועלת בכל המחשבים המחוברים למערכת ומשתמשים בפרוטוקול התקשורת.

מהות היכולת היא לדאוג שהתקשורת במערכת לא תהיה גלויה לכל מי שנתקל בה. היכולת מופעלת בכל פעם שישות מתכוונת להשתמש בפרוטוקול התקשורת על מנת לשלוח חבילה לישות אחרת. היכולת דואגת להצפין חלקים שחשוב שיהיו מוצפנים בחבילה.

אוסף התהליכים שהיכולת צריכה לבצע:

I. לקבל את החבילה מהיכולת שמתכוונת להשתמש בפרוטוקול התקשורת.
II. להצפין את כל החלקים שהוגדרו על ידי המערכת מראש כמכילים תכנים רגישים.
III. להעביר את החבילה ליכולת שתשלח אותה לישות המיועדת.

שליחת חבילה באמצעות פרוטוקול התקשורת

היכולת מופעלת כל הזמן. למעשה היא מבצעת את מטרתה רק ברגע שמשהו מיועד לשליחה.

מהות היכולת היא ליצור סדר ובקרה. היכולת היא פתרון לבלגן שיכול להיווצר אם כל יכולת תשלח חבילות על דעת עצמה.

אוסף התהליכים שהיכולת צריכה לבצע:

I. לבדוק האם קיימת חבילה שמיועדת לשליחה.
II. להבין למי צריך לשלוח את החבילה.
III. לשלוח את החבילה.

רשימת האובייקטים שהיכולת עושה בהם שימוש:

חבילה, תקשורת.

קבלה וניווט חבילה באמצעות פרוטוקול התקשורת

היכולת פועלת כל הזמן.

מהות היכולת לקלוט את כל החבילות שמיועדות להגיע לישות שבה היא פועלת (כל מחשב המחובר למערכת). לאחר מכן תעקוב היכולת אחר הכותרות של החבילה ועל פי חוקי הפרוטוקול תעביר את החבילה ליכולת שאמורה לטפל בה.

אוסף התהליכים שהיכולת צריכה לבצע:

I. קליטת החבילה.

II. זימון היכולת: פענוח מסרים.
III. מעקב אחר הכותרות של החבילה.
IV. העברת החבילה ליכולת שאמורה לקבל אותה.

רשימת האובייקטים שהיכולת עושה בהם שימוש:

חבילה, כותרות, תקשורת.

סריקת הבקשה בצד השרת

לפני שהשרת מתחיל לעבוד עם הבקשה הוא סורק אותה. הן מהיבטים בטחונים והן מהיבטים לוגיים, שכן המערכת מצפה לקלטים מסוימים. זוהי היכולת הראשונה שלמעשה מתעסקת עם תוכן הבקשה.

מהות היכולת היא לא לאפשר לשרת לעבוד על בקשה שלא עובדת בסטנדרטים ולהפריד בין תהליך הבדיקות לתהליך העבודה על הבקשה לשם הסדר והיעילות. (אין סיבה לעבוד על החלק הראשון בבקשה כשידוע שהחלק האחרון בה לא תקין).

אוסף התהליכים שהיכולת צריכה לבצע:

I. לקבל את הבקשה.
II. לזמן את היכולת שסורקת את הבקשה בהיבט הביטחוני.
III. לזמן את היכולת שבודקת האם הבקשה מכילה קלטים שהיכולת לא מצפה אליהם.

אוסף האובייקטים שהיכולת עושה בהם שימוש:

בקשה, קלט, פרוטוקול התקשורת.

סריקה ביטחונית של בקשה

מהות היכולת לבדוק האם יש תוכן מסוכן בבקשה.

אוסף התהליכים שהיכולת צריכה לבצע:

I. לזמן את היכולת שבודקת האם קיימים ניסיונות לביצוע sql injection שעלולים להדליף את תוכן מסד הנתונים או לשנות אותו.
II. בדיקה ממי התקבלה הבקשה. אם מדובר במחשב לא מזוהה לזמן את היכולת שבודקת האם קיים חשד לdos attack. כמו כן, על היכולת לדאוג שלא להעביר את הבקשה לשלב הבא (מניעה מזרים למערכת לנצל את משאביה).

רשימת האובייקטים שהיכולת עושה בהם שימוש:

בקשה, מסד נתונים, מחשב לא מזוהה.

בדיקת בקשה בשרת

היכולת מופעלת על ידי היכולת: "סריקת הבקשה בצד השרת".

מהות היכולת היא לבדוק האם קיימים קלטים שמראש אין סיבה לשרת לבדוק אותם שכן הוא כבר הודיע למשתמש כי הם מחוץ לטווח. יש לשים לב שהסיכוי שאכן יתקבל קלט מחוץ לטווח לא אמור להיות גבוה שכן יש הגבלה על כך עוד במחשב הלקוח לפני ששלח את הבקשה. הדבר בכל זאת יכול להתקיים במידה ותוך כדי שליחת הבקשה מצב המערכת הספיק להשתנות ועל כן הטווח שצוין אצל הלקוח כבר אינו עדכני. סיבה נוספת לקלט מחוץ לטווח היא שהמשתמש עקף את החסימה של קוד הלקוח.

אוסף התהליכים שהיכולת צריכה לבצע:

I. לעבור על התוכן של כל הקטגוריות בבקשה.
II. לבדוק בכל קטגוריה האם התוכן תואם את טווח האפשרויות שהשרת קבע לו.
III. במידה ולא, לא לקדם את הבקשה לשלב הבא ולהחזיר הודעה למשתמש כי הוא חרג מהטווח ושהקלט כבר לא פרקטי.

רשימת האובייקטים שהיכולת עושה בהם שימוש:

קלטים מחוץ לטווח, בקשה, קטגוריה.

פישוט הבקשה בצד השרת

לאחר שהבקשה עברה סריקה הדבר הראשון שעל השרת לעשות הוא לפרק אותה על מנת שיוקל עליו בהמשך כשינתח אותה ויקבל החלטה בדבר אופן מימושה.

מהות היכולת היא לספק ליכולות הבאות אחריה שיעבדו מול הבקשה, אותה באופן מסודר ונגיש.

אוסף תהליכים שהיכולת צריכה לבצע:

I. ראשית עליה לקבל את הבקשה מהיכולת סריקת הבקשה בצד השרת.
II. על היכולת לעבור על כל בקשה, לחלץ את כל הקלטים שסיפק המשתמש בכל קטגוריה, למיין אותם כשהתכלית היא נגישות ולשמור אותם במסד הנתונים. לדוגמה את ההגבלות שהמשתמש מתחייב אליהן בבקשה, תשמור במסד הנתונים בטבלה שאחראית על כך ועוד.

רשימת האובייקטים בהם היכולת עושה שימוש:

- מסד נתונים – היכולת כותבת את הנתונים שחילצה מהבקשה אל מסד הנתונים של השרת.
- אובייקטים נוספים: שרת, בקשה.

ניתוח הבקשה בצד השרת

לאחר פירוק הבקשה והכנסתה למסד הנתונים יהיה על השרת לנתח אותה ולקבל החלטה: האם ניתן לייצר ממנה הוראה או שעל המשתמש לבצע התפשרויות.

מהות היכולת היא לנתח את הבקשה לשם יצירת הוראה סופית בהמשך. יש להצליב את דרישות המשתמש עם הגבלות השרת והגבלות כל מחשב במערכת ולפעול לפי סדר העדיפויות של המערכת. לדוגמה: המשתמש דורש שמונה מוציאים לפועל אך מחוברים למערכת רק שישה.

אוסף התהליכים שהיכולת צריכה לבצע:

I. לקרוא את הבקשה ממסד הנתונים (התהליך מתבצע לאורך כל מימוש היכולת).

II. לעבור על כל הדרישות שציין המשתמש בבקשה.
III. לבדוק האם הבקשה של המשתמש עומדת בדרישות המפעיל. במידה ולא להחזיר תשובה למשתמש כי הבקשה לא יכלה לצאת לפועל הואיל והוא חרג מהגבלות המפעיל.
IV. למנות את מספר המחשבים בכיתה שעומדים בדרישות המצוינות בבקשה. במידה ולא קיימים מספר מחשבים כפי שביקש המשתמש לא להוציא את הבקשה לפועל ולהודיע לו כי קיימים רק x מחשבים פנויים שיכולים לטפל בבקשה.

<u>רשימת האובייקטים בהם היכולת עושה שימוש</u>:

בקשה, מסד נתונים, הוראה, הגבלות, דרישות, סדר עדיפויות, תקשורת.

<u>יצירת הוראה</u>

מהות היכולת היא לקחת את תוצאות ניתוח הבקשה שעשתה היכולת: "ניתוח הבקשה בצד השרת" (במידה והיכולת לא הודיע למשתמש כי עליו לבצע שינויים בבקשה) וליצור מהן הוראה שיצטרכו למלא המחשבים שנבחרו מהמערכת.

<u>אוסף תהליכים שהיכולת צריכה לבצע</u>:

I. בסוף הניתוח, היכולת תצוות לכל חלק בהוראה את מה שהוחלט כאופציה הטובה ביותר.
II. היכולת תיצור הוראה סופית - ההוראה למעשה אינה בנויה בגוף אחיד, היא מתפצלת בין המחשבים המחוברים למערכת ולמחשב המשתמש. היא מכילה מה על כל מחשב לעשות, מי המשתמש שאותו הוא משרת ומה ההתחייבויות שהקובץ הרץ מתחייב לעמוד בהן. כמו כן תשלח הוראה נפרדת למחשב הלקוח שתכיל את רשימת כל המחשבים שאמורים לספק את שרותם לו.

<u>רשימת האובייקטים בהם היכולת עושה שימוש</u>:

הוראה, פרוטוקול תקשורת, משתמש, מוציא לפועל, קובץ הרצה.

<u>קבלת ההוראה בלקוח ושליחת תיעוד לשרת</u>

מהות היכולת היא לקבל את ההוראה מהשרת הראשי, את הקבצים הדרושים ואת ההגבלות שעל המחשב לציית להן.

אוסף תהליכים שהיכולת צריכה לבצע:

I. לקבל את ההוראה מהשרת.
II. למיין ולשמור את ההגבלות, הקבצים והנתונים שהיכולת כוללת.
III. לעדכן את השרת כי ההוראה התקבלה בהצלחה וכי המחשב הנוכחי מתחיל בהכנות לקראת הרצת הקובץ.

רשימת האובייקטים בהם היכולת עושה שימוש:

הוראה, שרת ראשי, לקוח (משתמש), קובץ הרצה, פרוטוקול תקשורת.

יצירת הסביבה המבוקשת לקובץ הרץ

מהות היכולת היא ליצור לקובץ המיועד להרצה את הסביבה המתאימה לו על פי ההוראה שקיבלה מהשרת.

אוסף תהליכים שהיכולת צריכה לבצע:

I. היכולת מתבצעת אצל המוציא לפועל שהשרת בחר בו לספק את משאביו לטובת משתמש ששלח לו בקשה. ראשית על היכולת לקבל את הקובץ המיועד להרצה והקבצים הנלווים. היות והדבר נעשה כבר על ידי היכולת: "קבלת ההוראה בלקוח ושליחת תיעוד לשרת" יהיה על היכולת הנוכחית להשתמש בקבצים שקיבלה היכולת הנ"ל.
II. ליצור תיקיה חדשה (אם לא קיימת) על שם מה שיקבע המשתמש בבקשה שלו לשרת + הid של הבקשה (למניעת כפילויות), שהמיקום שלה לא יהיה כזה שיפריע למשתמש במחשב הנוכחי. (למשתמש אין שום יכולת לקבוע מה תהיה המחצה. כל מה שיכול המשתמש לעשות זה לקבוע את שם ההרצה הנוכחית והשרת ישתמש בשם זה על מנת לפתוח להרצה זו תיקייה חדשה במחצה שנקבעה מראש.)
III. העברת הקבצים לתיקיה החדשה.

רשימת האובייקטים בהם היכולת עושה שימוש:

קובץ הרצה, קבצים נלווים, בקשה, תיקיה.

הרצת לקוח באופן שאינו ידני

המערכת לא יכולה להתקיים ללא היכולת הזו, שכן זוהי מהות המערכת.

מהות היכולת היא לקבל קובץ הניתן להרצה ולהריץ אותו ללא התערבותו של משתמש בדיוק כפי שהקובץ היה רץ על ידי המשתמש עצמו.

אוסף תהליכים שהיכולת צריכה לבצע:

I. זימון היכולת: "יצירת הסביבה המבוקשת לקובץ הרץ".
II. לספק לקובץ סביבה שבה תוכל להריץ אותו. (לפתוח את שורת הפקודות ולהסתיר אותו כך שלא יהיה גלוי למשתמש במחשב הנוכחי).
III. היכולת תוציא לפועל את הפקודה שתגרום להרצת הקובץ.

רשימת האובייקטים בהם היכולת עושה שימוש:

מערכת הקבצים של מערכת ההפעלה, קובץ הרצה.

הפסקת כל פעילות של קובץ רץ

מהות היכולת היא להפסיק לצמיתות כל פעילות הקשורה להרצת קובץ.

אוסף תהליכים שהיכולת צריכה לבצע:

I. להרוג את התהליך שמריץ את הקובץ.
II. לעצור את כל היכולות שקשורות לקובץ הרץ ואחראיות לדבר הקשור בהרצתו.
III. למחוק את כל הקבצים השמורים במחיצה של ההוראה הנוכחית.
IV. לעדכן את השרת כי הקובץ לא רץ יותר על המחשב הנוכחי.

אובייקטים בהם היכולת עושה שימוש:

קובץ הרצה, שרת, תהליך שמריץ את הקובץ.

שליחה של תוצאות הקובץ לשרת

מהות היכולת היא לשלוח לשרת תיעוד בזמן אמת לגבי כל דבר הקשור לקובץ הרץ: פלט התכנית והחריגות שלה. בסוף ההרצה תשלח היכולת לשרת את קובץ הקלטים שאליו כתבה לאורך כל ההרצה.

אוסף תהליכים שהיכולת צריכה לבצע:

I. לשלוח לשרת את פלט הקובץ הרץ והחריגות שלו.
II. בסוף הריצה תשלח לשרת את הקבצים הסופיים הנחוצים.

רשימת האובייקטים בהם היכולת עושה שימוש:

קובץ רץ, פלט, חריגות, קבצים נלווים, חבילה.

שליחת תיעוד הקובץ הרץ מהשרת למשתמש

מהות היכולת היא לשלוח למשתמש את התיעוד של הקובץ הרץ שהשרת קיבל מכל אחד מהמחשבים המריצים. היכולת מבדילה בין מחשבים שהמשתמש ביקש לראות את התיעוד אחריהם לבין כאלה שלא. המשתמש יכול לבחור אם הוא רוצה את הפלט של הקובץ הרץ והחריגות שלו ואם הוא מעוניין בקבצים הנלווים.

אוסף התהליכים שהיכולת צריכה לבצע:

I. להתעדכן אחרי אילו מחשבים המשתמש מעוניין לעקוב.
II. להרכיב חבילה על פי תנאי הפרוטוקול שתכיל את התיעוד של כל מחשב.

רשימת האובייקטים שהיכולת עושה בהם שימוש:

תיעוד, קובץ רץ, מחשב מריץ, חבילה, פרוטוקול תקשורת.

ממשק המשתמש

המערכת מספקת יכולות רבות שהמשתמש יכול לנצל אותן.

מהות היכולת היא לספק ממשק שיוכל לקשר את יכולות המערכת למשתמש בה. כמו כן עליה לאפשר למשתמש להפעיל יכולות אלו כרצונו. הממשק שתספק יאפשר למשתמש לעקוב אחר הקובץ הרץ באופן יעיל ונוח. הממשק עצמו מורכב ממספר חלונות שהמשתמש יוכל לתמרן ביניהם: חלון להכנת בקשה ושליחתה לשרת, חלון שבו יוכל המשתמש לעקוב אחר הקובץ הרץ ועוד. כמו כן בממשק יהיה קיים סרגל בקרה שיכיל את הכפתורים שיוכל המשתמש להפעיל: שליחת הבקשה הנוכחית לשרת.

<u>אוסף תהליכים שהיכולת צריכה לבצע</u>:

I. קליטת נתונים הן מהמערכת והן מהלקוח: לחיצת כפתור, מידע חדש לגבי הקובץ הרץ ועוד.
II. עדכון כל מידע רלוונטי שהיא מקבלת בממשק: אם המשתמש בחר להציג חלון אחר היכולת תחליף לחלון המבוקש ועוד.

<u>רשימת האובייקטים בהם היכולת עושה שימוש</u>:

- מודול גרפי – היכולת עושה שימוש במודול גרפי על מנת לספק למשתמש gui שיוכל לתקשר בינו לבין המערכת.
- אובייקטים נוספים: ממשק גרפי, קובץ הרצה, חלון גרפיים, בקשה, סרגל בקרה, תקשורת.

<u>ממשק המפעיל</u>

המפעיל הוא זה שמנהל את מחשב השרת.

מהות היכולת היא לספק ממשק שיוכל לתקשר בין המפעיל לתכונות שהשרת מספק. בעזרת הממשק המפעיל יכול לפרט לשרת מהן ההגבלות שהוא בוחר להציב למערכת ועוד. כמו כן הממשק יציג למפעיל את המצב הנוכחי בכיתה. בדומה לממשק הלקוח גם ממשק המפעיל יכלול מספר חלונות: חלון שיציג את המצב הנוכחי של המערכת, חלון שיקלוט ממנו את ההגבלות והתנאים שהוא מציב במערכת וסרגל שבו יוכל לנהל את המערכת: להפסיק כל פעילות שלה ועוד.

<u>אוסף תהליכים שהיכולת צריכה לבצע</u>:

I. קליטת נתונים הן מהמערכת והן מהמפעיל.
II. עדכון כל מידע רלוונטי שהיא מקבלת בממשק.

רשימת האובייקטים בהם היכולת עושה שימוש:

- מודול גרפי – היכולת עושה שימוש במודול גרפי על מנת לספק למפעיל gui שיוכל לתקשר בינו לבין המערכת.
- מסד נתונים – היכולת נעזרת במסד הנתונים בכדי להציג בממשק את המצב הנוכחי של המערכת למפעיל.
- אובייקטים נוספים: חלון גרפי, סרגל בקרה.

הצגת תמונת מצב של המערכת למפעיל

מהות היכולת היא לאפשר למפעיל המערכת לדעת מה מצבה הנוכחי במעבדה.

אוסף התהליכים שהיכולת צריכה לבצע:

I. לקבל את הממצאים מהמחשבים המחוברים למערכת בעזרת פרוטוקול התקשורת.
II. לספק את הממצאים ליכולת שמטפלת בממשק המפעיל. היכולת תעביר לממשק רק את הממצאים שהמפעיל ביקש להציג.

הפסקת פעילות- לקוח

למשתמש יש את האפשרות ללחוץ על כפתור הפסק.

מהות היכולת היא לאפשר למשתמש לעצור את הרצת הקובץ ולמעשה לעצור את הטיפול בבקשה הנוכחית שלו.

אוסף תהליכים שהיכולת צריכה לבצע:

I. להיוודע שהמשתמש לחץ על הכפתור הנ"ל ולבדוק האם יש לו בקשה פעילה במערכת.
II. לשאול את המשתמש האם הוא בטוח בצעד שהוא ביקש לעשות.
III. שליחה לשרת שהמשתמש מבקש להפסיק כל טיפול בבקשה הנוכחית.
IV. שליחת פקודה מהשרת אל כל המחשבים המעורבים להפסיק כל פעילות הקשורה לבקשה הספציפית הזו והסרת הבקשה מהטבלה בבסיס הנתונים

המכילה את הבקשות הנוכחיות (המחשבים המעורבים עושים זאת על ידי היכולת: "הפסקת כל פעילות של קובץ רץ").

רשימת האובייקטים בהם היכולת עושה שימוש:

- תקשורת – היכולת עושה שימוש בתקשורת על מנת להעביר את ההודעה על ביטול הן מהמשתמש לשרת והן מהשרת למחשבים המעורבים.
- אובייקטים נוספים: כפתור, קובץ הרצה, בקשה, שרת, מוציאים לפועל.

התחלה מחדש

למשתמש יש את האפשרות ללחוץ על כפתור התחלה מחדש.

מהות היכולת היא לאפשר למשתמש להפסיק את פעילות הבקשה הנוכחית ולהפעיל אותה מחדש.

אוסף תהליכים שהיכולת צריכה לבצע:

I. להיוודע שהמשתמש לחץ על הכפתור הנ"ל.
II. לשאול את המשתמש האם הוא בטוח בצעד שהוא ביקש לעשות.
III. שליחה לשרת שהמשתמש מבקש להתחיל את הבקשה שבחר מחדש.
IV. השרת נעזר ביכולת הפסקת פעילות ומפסיק כל פעילות של הבקשה הנוכחית.
V. השרת מתייחס לבקשה כאילו כרגע קיבל אותה (במידה והמשתמש ביצע בה שינויים- שינה את תוכן הקובץ להרצה או עדכן את קובץ הקלט וכו').

הפסקת פעילות- שרת

למנהל השרת יש את האפשרות ללחוץ על כפתור הפסק.

מהות היכולת היא למנוע כל סוג של פעילות במערכת.

אוסף תהליכים שהיכולת צריכה לבצע:

I. להיוודע שהמשתמש לחץ על הכפתור הנ"ל.

II. השרת נעזר ביכולת הפסקת פעילות ומפסיק את כל הפעילות של כל הבקשות הפעילות כרגע.
III. השרת דוחה כל בקשה חדשה המתקבלת מהמשתמשים בטענה שהמערכת בהשהיה.

רשימת האובייקטים בהם המערכת עושה שימוש:

כפתור, בקשה, השהיה.

שליחת נתונים מהמחשב המריץ אל השרת

מהות היכולת היא לשלוח את הפלט והתעופות של הקובץ הרץ אל השרת המרכזי. היכולת פועלת על כל מוציא לפועל.

אוסף תהליכים שהיכולת צריכה לבצע:

I. לקבל את הממצאים בנוגע למחשב הנוכחי מהיכולת שמריצה את הקובץ.
II. לסדר את הממצאים כך שיתאימו לתנאי הפרוטוקול.
III. לשלוח את התוצאה לשרת המרכזי.

רשימת האובייקטים בהם היכולת עושה שימוש:

נתונים, ממצאים, פרוטוקול, תוצאה, תקשורת.

שמירת המצב הנוכחי של המערכת במסד הנתונים של השרת

מהות היכולת היא לאגור את הנתונים שיכולים להיות הכרחיים לטובת ייעול המערכת ושיכולים להיות רלוונטיים הן לשרת, הן למפעיל והן למשתמשים.

אוסף תהליכים שהיכולת צריכה לבצע:

I. לקבל את הממצאים של כל מחשב המחובר למערכת בעזרת פרוטוקול התקשורת.
II. לתרגם אותם אל תוך מסד הנתונים של השרת.

רשימת האובייקטים שהמערכת עושה בהם שימוש:

נתונים, מסד נתונים.

טיפול במחשב מריץ שמתנתק

מהות היכולת לתת מענה להוראה שאחד המחשבים שמטפלים בה מתנתק.

אוסף תהליכים שהיכולת צריכה לבצע:

I. להיוודע שאחד המחשבים המריצים התנתק.
II. להודיע למשתמש כי המחשב התנתק ולשאול אותו האם הוא מעוניין שהמערכת תמצא לו מחליף.
III. במידה וכן, לפנות ליכולת שמנתחת בקשות ולשאול האם יש לה מחשב שיכול להתמודד עם הבקשה הנוכחית כל זמן מסוים.
IV. לעדכן את בסיס הנתונים.

טיפול במחשב משתמש המריץ בקשה שמתנתק

מהות היכולת להפסיק את הפעילות של מחשבים שמטפלים בהוראה של בקשה שמחשב של משתמש שלח לשרת.

אוסף תהליכים שהיכולת צריכה לבצע:

I. להיוודע שמחשב שבקשה שלו כרגע מטופלת במערכת התנתק.
II. לזמן את היכולת שמפסיקה כל פעילות בדבר בקשה מסוימת.

רשימת האובייקטים בהם היכולת עושה שימוש:

הוראה, בקשה.

פרק ד'- פרק העיצוב

תיאור הארכיטקטורה של המערכת המוצעת

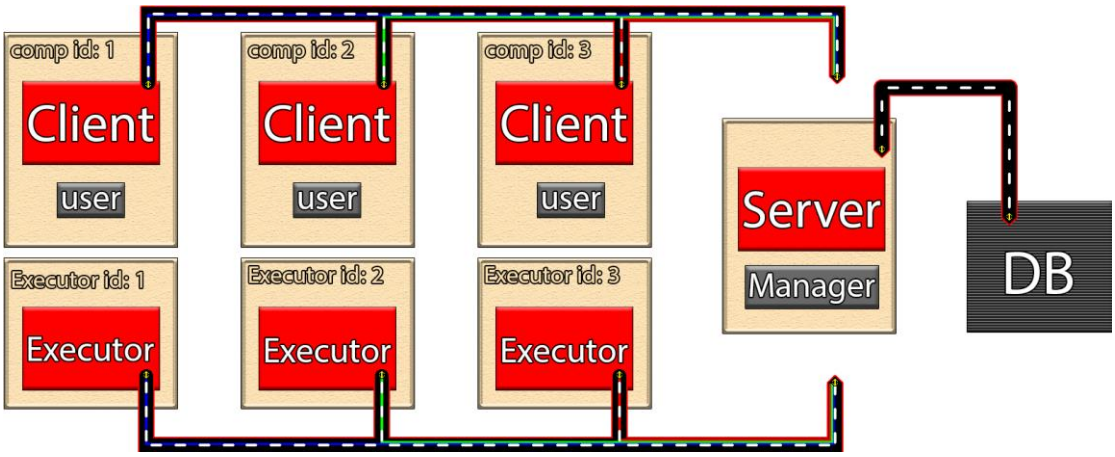
תיאור החומרה

מערכת הפעלה windows, מעבד, זיכרון חיצוני ופנימי לכל מחשב במערכת. מסד נתונים שנפחו בהתאם לממדי המעבדה בה המערכת מותקנת. רכיבי רשת שיוכלו לספק תקשורת בין המחשבים המחוברים למערכת. אופציונאלי: מערכת וירטואלית (virtual box oracle) המריצה מערכת הפעלה windows (ניתן מטעמי אבטחה להפעיל את המוציאים לפועל מתוך מכונה וירטואלית).

רכיבים שונים והקשרים ביניהם

המחשבים המחוברים למערכת מחולקים לשתי קבוצות: מחשבים שמופעלים על ידי המשתמשים הרגילים ומוציאים לפועל. על כל אלה מנצח מפעיל המערכת שלו שליטה ישירה על השרת המרכזי. השרת הראשי מנהל בסיס נתונים, לכן כל רכיב שמעוניין לתקשר עם בסיס הנתונים חייב לעבור קודם דרך השרת הראשי. בסיס הנתונים מכיל את כל המידע הנחוץ למערכת הן לטווח הקצר והן לארוך.

תרשים המתאר את הרכיבים השונים והקשרים ביניהם



תיאור הטכנולוגיה הרלוונטית

המערכת באה לענות על צורך שקיים בקרב מפתחים רבים. הטכנולוגיה כיום יכולה לתת מענה לצורך הזה בשונה מן העבר היות והמחשבים היום יכולים להריץ מספר תכניות במקביל מבלי לפגוע בביצועיהם באופן ניכר.

תיאור מודולים בהם נעשה שימוש

Client UI (ממשק הלקוח)

המחלקה אחראית לכל דבר הקשור לממשק של המשתמש.

ראשית, היא בונה למשתמש את הממשק הגרפי. לאחר מכן היא מוציאה לפועל את הדברים שהמשתמש מבצע באמצעות הממשק. שומרת את הקלט שהוא מכניס במקומות המתאימים לו בקוד, מציגה לו את המסך שהוא מבקש להציג, קוראת לחלקים בקוד שמוציאים לפועל משימות שביקש המשתמש לבצע באמצעות לחיצה על כפתורים ועוד.

start:

טענת כניסה: הפעולה לא מקבלת דבר.

טענת יציאה: הפעולה בונה את הממשק הגרפי של המשתמש.

debug box:

טענת כניסה: הפעולה מקבלת הודעה.

טענת יציאה: הפעולה מדפיסה את ההודעה שהתקבלה בdebug box.

Client Request Validation (בדיקת תקינות הבקשה בלקוח)

המחלקה אחראית לוודא שהבקשה שהכין המשתמש עומדת בסטנדרטים הראשוניים שהציב השרת המרכזי לכל הלקוחות המחוברים למערכת. למעשה, המחלקה דואגת להרצת thread שכל הזמן בודק האם הבקשה הנוכחית המוצגת בממשק המשתמש עומדת בסטנדרטים שהציב השרת. במידה ולא דואגת להודיע למשתמש מה לא תקין בבקשה שלו.

manager:

טענת כניסה: הפעולה לא מקבלת דבר.

טענת יציאה: הפעולה עוברת על רשימת הקלטים שכרגע נמצאים בשדות הקלט בחלון הבקשה של המשתמש. מדפיסה ליד כל קלט את טווח הקלטים האפשריים הרלוונטיים לו. אם מוצאת קלט שמחוץ לטווח, מזמנת את unavailable input עם מזהה השדה הבעייתי.

unavailable input:

טענת כניסה: הפעולה מקבלת את מזהה השדה שבו הקלט לא תקין.

טענת יציאה: הפעולה מוחקת את תוכנו ומודיע למשתמש באמצעות הdebug box מדוע הקלט לא אפשרי.

Client Communication (תקשורת הלקוח)

המחלקה אחראית לכל דבר הקשור לתקשורת עם ישויות אחרות במערכת.

היא מבדילה בין שני מיקרים: קבלת הודעה מהלקוח הנוכחי לשם שליחתה לישות מסוימת אחרת המחוברת למערכת וקבלת הודעה מישות המחוברת למערכת לשם העברתה לחלק בקוד שאליו היא מיועדת. אם קיבלה הודעה לשליחה תעביר אותה קודם למחלקה שדואגת להצפין אותה ואז תשלח אותה, אם קיבלה הודעה בשביל

הלקוח הנוכחי תעביר אותה למחלקת פיענוח ואז תעביר אותה לחלק שאליו היא מיועדת.

send:

טענת כניסה: הפעולה מקבלת הודעה ויעד.

טענת יציאה: הפעולה מוודאת שההודעה עוברת דרך מודול ההצפנה. לאחר מכן שולחת את ההודעה בהתאם לדרישה.

receive:

טענת כניסה: הפעולה לא מקבלת דבר.

טענת יציאה: הפעולה מעבירה את ההודעה אל מודול הפיענוח. לאחר מכן, לפי הכותרות שלה וחוקי הפרוטוקול מעבירה אותה ליעד שלה בקוד.

Encryption (הצפנה)

מקבלת הודעה הבנויה על פי חוקי הפרוטוקול ומצפינה אותה כנדרש. ראשית, היא מזהה את סוג ההודעה (לפי הכותרות של הפרוטוקול) ולאחר מכן מצפינה את התוכן שלה על פי כללים שהוגדרו מראש.

identify:

טענת כניסה: הפעולה מקבלת הודעה הבנויה לפי חוקי הפרוטוקול.

טענת יציאה: הפעולה עוברת על הכותרות של ההודעה ומזהה מה סוג התוכן. מזמנת את encrypt בכל פעם עם חלק מהתוכן ושיטת ההצפנה הרלוונטית לו. לאחר מכן מחזירה את ההודעה.

encrypt:

טענת כניסה: הפעולה מקבלת הודעה (תוכן) ושיטת הצפנה.

טענת יציאה: הפעולה מצפינה את התוכן על פי שיטת ההצפנה שהתקבלה ומחזירה את ההודעה.

Decryption (פיענוח)

עובדת בדיוק כמו המחלקה "Encryption (הצפנה)" רק שבמקום להפעיל את פעולת ההצפנה המתאימה מפעילה את פעולת הפיענוח המתאימה.

Server Communication (תקשורת השרת)

פועלת לפי אותו רעיון שהמחלקה "Client Communication (תקשורת הלקוח)" פועלת. הפעולה receive מטפלת בכותרות שונות, היות והשרת והלקוח לא מקבלים את אותן החבילות.

Executor Communication (תקשורת המכונה הוירטואלית)

פועלת לפי אותו רעיון שהמחלקה "Client Communication (תקשורת הלקוח)" פועלת. הפעולה receive מטפלת בכותרות שונות, היות והמכונה הוירטואלית והלקוח לא מקבלים את אותן החבילות.

Request Dismantling (פירוק בקשה)

מקבלת בקשה של משתמש, מפרקת אותה ומכניסה אותה אל תוך מסד הנתונים. לאחר מכן מסמנת את הבקשה במסד הנתונים כ"עברה תהליך פירוק".

dismantle:

טענת כניסה: הפעולה מקבלת בקשה כפי שהתקבלה מהמשתמש.

טענת יציאה: הפעולה מפרקת את הבקשה ומכניסה את תוכנה אל תוך מסד הנתונים. כשסיימה, מסמנת כי הבקשה סיימה את תהליך הפירוק.

Data Base (בסיס נתונים)

כל התעסקות כלשהי עם בסיס הנתונים עוברת דרך המחלקה הזו. בעזרת שפת הSQL היא מתקשרת עם בסיס הנתונים. SQL לא תופיע בשום מקום אחר בקוד מלבד במחלקה הזו- גם הפעולות של המחלקה לא מקבלות קוד SQL. השימוש במחלקה נעשה על ידי קריאה לפעולות שלה עם פרמטרים מכוונים.

create table:

טענת כניסה: הפעולה מקבלת שם של טבלה, רשימת המכילה את השורות ורשימה המכילה את הטורים של הטבלה.

טענת יציאה: הפעולה יוצרת את הטבלה במסד הנתונים לפי הפרמטרים שקיבלה.

query:

טענת כניסה: הפעולה מקבלת הוראה לביצוע שאילתה הכוללת את שם הטבלה, שם העמודה, השורות המבוקשות ותנאי מסויים.

טענת יציאה: הפעולה מבצעת את השאילתה אל מול מסד הנתונים ומחזירה את התוצאות.

update:

טענת כניסה: הפעולה מקבלת הוראה לבצע סוג של עדכון שמוכר על ידי המערכת ואת הנתונים שלו. יש אפשרות גם לקבל שם של טבלה, שם של עמודה, שם של שורה ותנאי מסוים.

טענת יציאה: הפעולה מעדכנת את הערכים בבסיס הנתונים על פי הפרמטרים שקיבלה.

Server request Validation (בדיקת תקינות בקשה בשרת)

מבצעת את אותם הדברים שמבצעת המחלקה Client Request Validation (בדיקת תקינות בקשה בלקוח) רק שבמקום לדאוג להדפסת התקלות בבקשה בממשק המשתמש תדאג לשלוח אותן למשתמש ולסמן את הבקשה כחסרת יכולת מימוש. חיונית לאבטחה חזקה יותר ומונעת אפשרות של שינוי מצב במערכת בזמן שנבדקה אצל הלקוח ועד שהתקבלה לשרת שגרם לה להיות לא תקינה.

unavailable input:

טענת כניסה: הפעולה מקבלת את מזהה השדה שבו הקלט הלא תקין.

טענת יציאה: הפעולה שולחת הודעה למשתמש שיצר את הבקשה שמכילה את הסיבה לכך שלא ניתן להוציא את הבקשה לפועל. מזמנת את הפעולה unavailable request.

unavailable request:

טענת כניסה: הפעולה מקבלת את הבקשה הלא תקינה.

טענת יציאה: הפעולה מעדכנת בבסיס הנתונים שהבקשה לא תקינה ומפסיקה כל התעסקות הקשורה לה.

Command Constructor (מרכיב הוראה)

תכלית המחלקה היא בניית הוראה מבקשה שהמחשבים שנקבעו במערכת יצטרכו לבצע.

ההוראה תהיה אולטימטיבית באספקטים הבאים: התאמה הבקשה למחשבים במערכת שיכולים לספק את המשאבים שהיא דורשת, ידידותיות למשתמשים (תעדיף לנצל מחשב ללא משתמש על פני מחשב עם) ועוד.

במקרה של סתירה בין מה שדורשת הבקשה לבין מה שמשאבי המערכת יכולים לספק, תעצור המחלקה כל התעסקות נוכחית ועתידית של הבקשה ותחזיר הודעה ללקוח שתכיל את הסיבה לכך שלא ניתן לטפל בבקשה שלו (הכוונה בסתירה הנוכחית היא כזו שלא ניתן לאתר ב Server/Client request Validation, אלא רק כזו שהשרת יכול להיתקל בה בחישובים המעמיקים הנוכחיים, כמו כן זהו הרגע שבו נוצרת ההוראה ובאופן תיאורטי יכול להתקיים שינוי במערכת מאז הבדיקה האחרונה של הבקשה).

find ultimate computers to run:

טענת כניסה: הפעולה מקבלת בקשה.

טענת יציאה: הפעולה רצה על הבקשה, בכל קטגוריה יוצרת שתי רשימות: האחת מכילה את המחשבים שיכולים לעמוד בקטגוריה הנוכחית והשנייה מכילה את הדירוג שלהם (ככל שהמחשב קרוב יותר לתחילת הרשימה כך הוא מתאים יותר לטיפול בקטגוריה). לאחר מכן מוצאת מיהם המחשבים שהכי כדאי להריץ עליהם את הבקשה. אם לא מוצאת מספיק מחשבים להריץ עליהם את הבקשה מודיע על כך למשתמש (יחד עם הפרמטרים הבעייתיים בבקשה), מסמנת את הבקשה כבלתי ניתנת לטיפול ועוצרת כל התעסקות הקשורה לה.

create command:

טענת כניסה: הפעולה מקבת את הנתונים של הבקשה על ידי תכונות המחלקה.

טענת יציאה: הפעולה דואגת לבניית עצם מסוג הוראה מהנתונים ש find ultimate computers to run מצא.

command (הוראה)

המחלקה מייצגת הוראה.

to protocol:

טענת כניסה: הפעולה לא מקבלת דבר.

טענת יציאה: הפעולה הופכת את ההוראה הנוכחית למחרוזת על פי חוקי הפרוטוקול ומחזירה אותה.

from protocol:

טענת כניסה: (פעולה אבסטרקטית) הפעולה מקבלת מחרוזת המייצגת עצם מסוג הוראה ובנויה על פי חוקי הפרוטוקול.

טענת יצאיה: הפעולה מחזירה עצם מסוג הוראה לפי המחרוזת שקיבלה.

Command Manager (מנהל בקשה)

אחראי על כל דבר הקשור להוראה במחשב לקוח.

דואג להרצת הקובץ להרצה, שולח את המידע והנתונים לגבי הקובץ הרץ לשרת, בודקת האם ההוראה לא חורגת את הגבולות שהיא נושאת עמה ועוד.

send command to executor:

טענת כניסה: הפעולה לא מקבלת דבר.

טענת יציאה: הפעולה שולחת אל הvm את ההוראה.

receive data from executor:

טענת כניסה: הפעולה לא מקבלת דבר.

טענת יציאה: הפעולה מקבלת נתונים אודות ההוראה במוציא לפועל. שומרת אותם באופן מסודר בתכונות המחלקה.

send data to server:

טענת כניסה: הפעולה לא מקבלת דבר.

טענת יציאה: הפעולה שולחת לשרת, לפי הפרוטוקול, את כל הנתונים הנוכחיים שהתקבלו מן המוציא לפועל לגבי ההוראה.

stop command:

טענת כניסה: הפעולה לא מקבלת דבר.

טענת יציאה: הפעולה דואגת להפסקת כל פעילות הקשורה להוראה הנוכחית.

validation checker:

טענת כניסה: הפעולה לא מקבלת דבר.

טענת יציאה: הפעולה מוודאת אל מול ההגבלות שההוראה נושאת והנתונים שהתקבלו מהvm שאין חריגה. אם מגלה חריגה מעדכנת את השרת על כך.

Executor manager (מנהל המוציא לפועל)

המחלקה מנהלת את תהליך הרצת הקובץ וכל מה שקשור אל תהליך זה.

create environment:

טענת כניסה: הפעולה לא מקבלת דבר.

טענת יציאה: הפעולה יוצרת תיקייה חדשה ושומרת בה את כל הקבצים שמכילה ההוראה.

setup std:

טענת כניסה: הפעולה לא מקבלת דבר.

טענת יציאה: הפעולה משנה את הstdin של מערכת ההפעלה לקובץ הקלט, את הstdout לקובץ שמטרתו להכיל את פלט התוכנית ואת הstderr לקובץ שיכיל את החריגות של הקובץ הרץ.

run file:

טענת כניסה: הפעולה לא מקבלת דבר.

טענת יציאה: הפעולה מריצה את הקובץ להרצה.

file supervision:

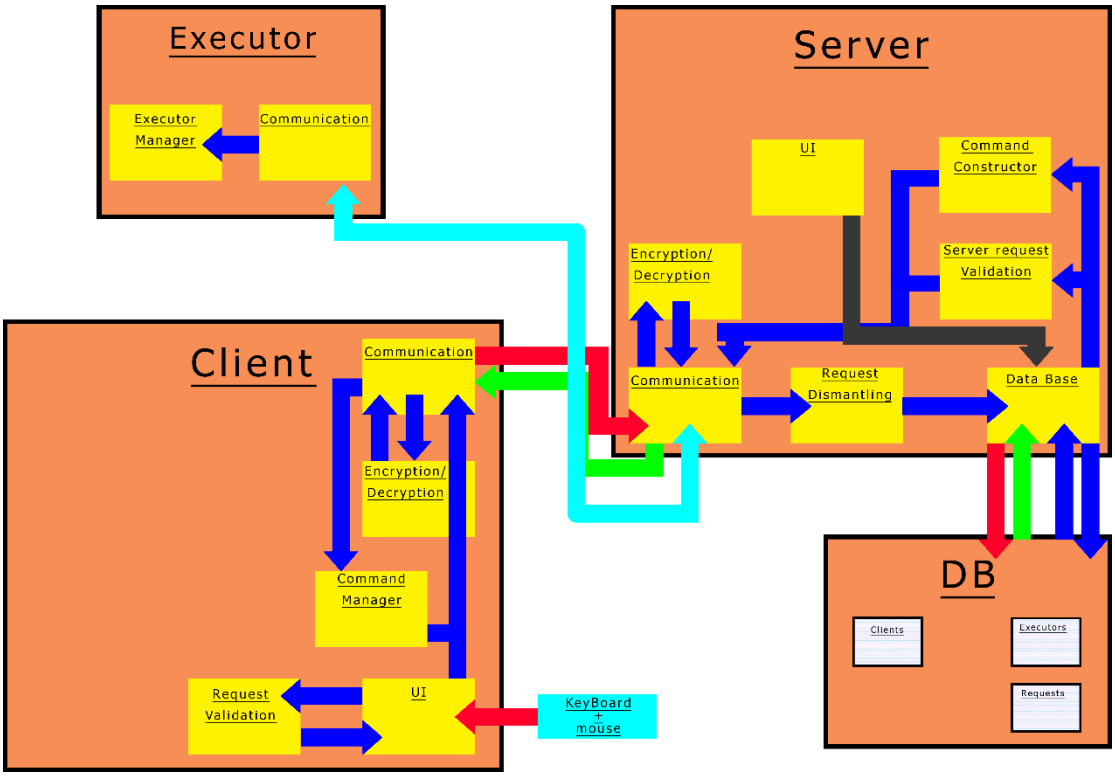
טענת כניסה: הפעולה לא מקבלת דבר.

טענת יציאה: הפעולה בודקת את התנהגות הקובץ וכמות המשאבים שהוא צורך. הפעולה מעדכנת את קוד הלקוח מחוץ לvm בדבר כל הנתונים שהיא מחלצת.

system modification:

טענת כניסה: הפעולה מקבלת את הערכים שהיא מתבקשת לשנות בvm.

טענת יציאה: הפעולה מעדכנת את הערכים.



תיאור סביבת הפיתוח

שפת התכנות שנבחרה לכתיבת הפרויקט

- Python
- MySQL
- windows

כלי הפיתוח הנדרשים לפיתוח

- מעבדה עם מחשבים חזקים מספיק על מנת להריץ עליהם את המערכת.
- רשת מקומית שכל המחשבים במעבדה מחוברים אליה.
- Code editor: PyCharm.

פירוט הסביבה והכלים הנדרשים לבדיקות

- מנהל המשימות למעקב אחר הthreads של התכנית ועוד.
- Wireshark לסריקת התעבורה של המערכת.
- File explorer לבדיקת המצב הנוכחי של קבצי המערכת.
- אופציונאלי : Oracle virtual machine, מריצה win10.

תיאור האלגוריתמים המרכזיים בפרויקט

הרצת קובץ במחשב מרוחק

ניסוח וניתוח של הבעיה האלגוריתמית

הרצת קובץ על מחשב מרוחק מהווה סיכון אבטחתי רציני ביותר. תוקף/עובד ממורמר שמתחבר למערכת יכול להפיץ וירוסים בקלות למספר רב של מחשבים ללא בקרה ולגרום לנזק רב.

תיאור אלגוריתמים קיימים לפתרון הבעיה

I. בקרה במוציא לפועל– המפעיל יוכל לעקוב אחר השפעות של הקובץ על המוציאים לפועל ובמקרה של חשד לנתק את המשתמש מהמערכת באזהרה.
II. הרצת הקובץ במערכת וירטואלית – המוציא לפועל שמריץ את הקובץ יריץ אותו במערכת הפעלה וירטואלית שתהיה פתוחה במחשב, כך לקובץ לא תהיה גישה למערכת ההפעלה הראשית של המחשב וכל פגיעה במערכת הווירטואלית היא זמנית.

הפנייה למקורות רלוונטיים

https://en.wikipedia.org/wiki/Virtual_machine

סקירת הפתרון הנבחר

הרצת הקובץ במערכת וירטואלית – ברגע שהקובץ רץ במערכת וירטואלית יש הפרדה מוחלטת בין הקובץ הרץ לבין מערכת ההפעלה של הלקוח. למעשה שני הצדדים מרוויחים: מצד אחד הקובץ הרץ מקבל מערכת הפעלה משלו והוא יכול לבצע כל העולה על רוחו מבלי להיות כפוף להגבלות כלשהן. מצד אחר המשתמש של המחשב הנוכחי (שעליו הקובץ מיועד לרוץ) לא נפגע מפעולותיו של הקובץ (אין שינוי לא רצוני במערכת הקבצים, דברים שהקובץ הרץ עושה לא ישנו את התצוגה הנוכחית ועוד). יתר על כן, גם אם תופעל בקרה ישירה על הקובץ הרץ במידה והוא לא ירוץ במערכת וירטואלית, הבקרה לעולם לא תוכל באמת להגביל אותו שכן קיים ארסנל אסטרונומי של פעולות היכולות להפריע לפעילות המשתמש שהקובץ יכול לבצע. כמו כן, פיתוח בקרה כזו יצרוך זמן רב שיכול להיות מושקע בפיתוח תכונות אחרות למערכת.

איזון המערכת

ניסוח וניתוח של הבעיה האלגוריתמית

המשאבים שמספקת המערכת מתחלקים בין כל המחשבים בה. הוראה לא יכולה להתפצל בין כמה מחשבים. כלומר, כמה מחשבים יכולים לטפל באותה הוראה אך מחשב לא יכול לטפל בחצי הוראה. כאן נובעת הבעיה. בכל רגע שנוצרת הוראה חדשה השרת צריך לבחור באילו מחשבים להריץ אותה. השרת צריך לקחת בחשבון פרמטרים רבים: מהן ההגבלות שמגיעות עם ההוראה (מה אחוז הcpu שהיא מבקשת לצרוך, מה כמות הזיכרון הפנימי ועוד), מה כל מחשב במערכת מוכן לספק, מהן ההגבלות שהציב המפעיל למערכת, מה המצב הנוכחי של המערכת (כמה בקשות מורצות בכל מחשב, אילו מחשבים מאוישים ואילו לא- כפתור שאם יהיה לחוץ המחשב ייחשב כלא מאויש ועוד). אם השרת לא היה מצוות בחוכמה את ההוראות למחשבי ההרצה המערכת לא הייתה מנוצלת ברמה המקסימלית שלה ואולי אפילו מגיעה לאחוזי נצילות בודדים.

תיאור אלגוריתמים קיימים לפתרון הבעיה

I. שימוש באלגוריתמים קיימים בנושא load – balancing.
II. פיתוח אלגוריתם ייחודי שיהיה ניתן להלביש אותו על המערכת הנוכחית.

הפנייה למקומות רלוונטיים

https://pdfs.semanticscholar.org/0c23/421ce8ae628ba70ccccbca09263db67d70ba.pdf

סקירת הפתרון הנבחר

כיוון שלא נמצא אלגוריתם load –balancing שניתן להלביש על המערכת, הפתרון שנבחר יכלול פיתוח אלגוריתם שיקבל השראה מהאלגוריתמים הקיימים בנושא, כך שעוד בפיתוחו תינתן הדעת בעניין ארכיטקטורת המערכת.

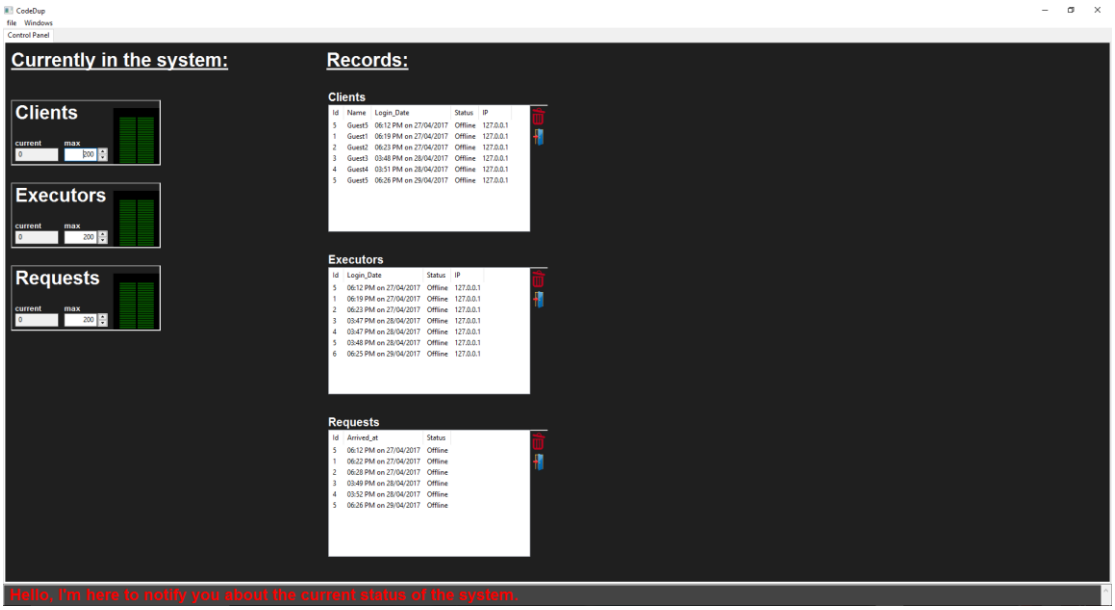
ראשית, האלגוריתם זקוק לנתונים ספציפיים לגבי משאביו ויכולותיו של כל מחשב במערכת. כיוון שהנתונים הללו תלויים בפרמטרים ייחודים למערכת הנוכחית יהיה על האלגוריתם להשיגם בכוחות עצמו. לשם כך, עם התחברות מחשב למערכת יבצע עליו השרת מספר בדיקות שיבדקו את יכולות המחשב הפרקטיים בקטגוריות הבאות:

- יכולת חישובית לוגית – כמה המחשב יעיל כשמדובר אך ורק בעבודה מול המעבד וההתנהלות שלו עם הזיכרון הפנימי..
- התעסקות עם זיכרון – כמה המחשב יעיל כשמדובר בכתיבה וקריאה מהזיכרון החיצוני.
- תקשורת – עד כמה המחשב יעיל בהתעסקות עם האינטרנט והתקשורת.
- לאחר שהאלגוריתם ישיג את הנתונים הללו הוא ישתמש בנוסחאות מוכנות מראש, בלוגיקה ובהעדפות שהוגדרו במערכת על מנת לקבוע אילו מחשבים יטפלו באילו הוראות.

תיאור מסכי הפרויקט

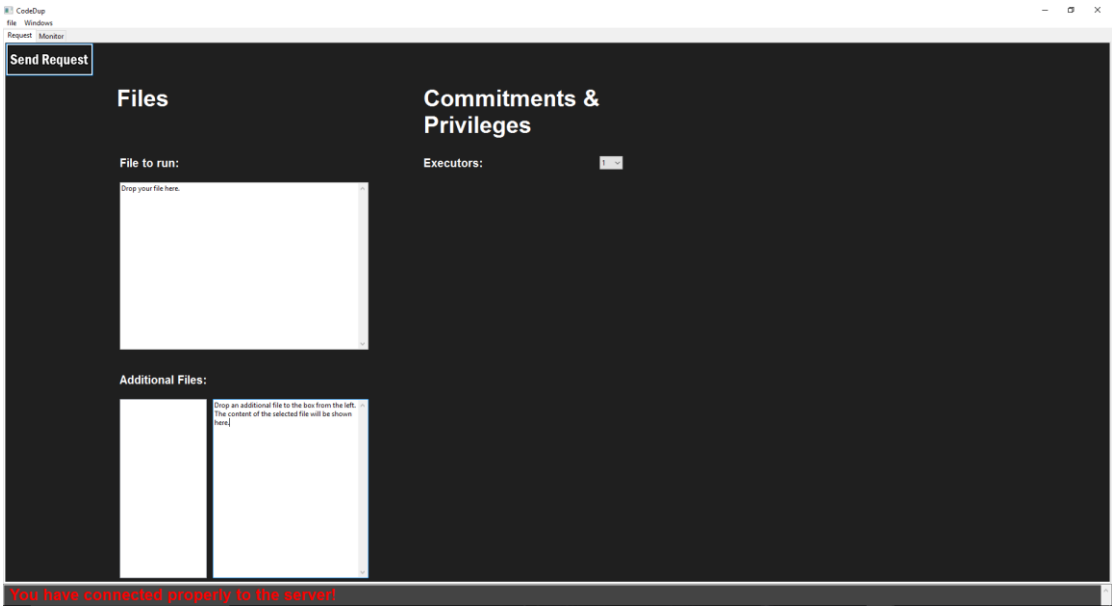
מסך הבקרה של המפעיל

מסך זה מאפשר למפעיל לעקוב אחר המצב הנוכחי של המערכת ולהיחשף לתיעוד מפעילויות ישנות. כמו כן למפעיל ניתנת היכולת לבצע פעולות כדי להגביל ולשנות את המצב הנוכחי.



מסך הרכבת הבקשה של המשתמש

מסך זה מאפשר ללקוח להרכיב בקשה ובסוף לשלוח אותה לשרת.



מסך המעקב של המשתמש

במסך זה יכול המשתמש לעקוב אחר הבקשה שלו כפי שהיא מורצת אצל כל אחד מן המוציאים לפועל.

תיאור מבני הנתונים

פירוט מבני הנתונים

מסד נתונים sql, קבצי טקסט, עצמים.

פירוט מאגרי המידע של המערכת

- **קבצי טקסט:** במקרים בהם המערכת מעוניינת לתעד מצב מסוים או לשמור היא תיעזר בקבצי טקסט.
- **עצמים:** המערכת תעשה שימוש בעצמים כמו רשימות ומילונים על מנת לאגור מידע שנעשה בו שימוש לעיתים תכופות. כמו כן היא תיעזר ביכולת של השפה לבצע עליהם פעולות מסוימות (ריצה, מיון וכו'). בנוסף, היא

תעשה שימוש בעצמים עם תכונות מיוחדות על מנת לשמור על הסדר והפשטות.

- **מסד הנתונים** : System Data :
  - כותרת : **Clients**

מספר הזיהוי של הלקוח	שם המשתמש של הלקוח	תאריך ההתחברות שמערכת	מצב (פעיל/לא פעיל)	כתובת ip
1				
2				
...				

  - כותרת : **Executors**

מספר הזיהוי של המוציא לפועל	תאריך ההתחברות	מצב (פעיל/לא פעיל)	כתובת ip
1			
2			
...			

  - כותרת : **Requests**

מספר הזיהוי של ההוראה	תאריך הוצאתה לפועל במערכת	מצב (פעילה/לא פעילה)	מה מספר הזיהוי של הבקשה שממנה נוצרה	...
1				
2				
...				

פרק ה'- הקוד

יצירת מסך המוניטור בשביל הלקוח

```
# Create a list of all the ids of the executors monitors as pairs,
# sorted.
self.pairs = self.executors_monitors.keys()
self.pairs.sort()

# Create a tree out of the ids as described in the Pair class.
self.monitors_tree = Pair()
self.monitors_tree.create_tree(self.pairs)

# Set the parent of the main splitter / monitor as the monitors panel.
self.monitors_tree.parent = self.monitors_panel

# Create the monitors and the splitters.
self.monitors_tree.vlr(self.executors_monitors, self.recorder)

# Split the splitters.
self.monitors_tree.lrv(True)

# Add to the main sizer of the monitors panel the monitors tree.
self.monitors_panel_sizer.Add(self.monitors_tree.value, 1,
wx.EXPAND)

# Call the Layout method to construct the monitors tree on the
# monitors panel graphically.
self.Layout()
```

```
from math import floor
import wx
from Monitor import Monitor


class Pair():
    """
        This class represents a binary tree. This tree is meant to be as follow:
                    pair
                pair        pair
            pair    pair    pair    pair
                    etc.
          id  id  id   id  id  id  id  id

        The value of each pair is a splitter which its parent is the splitter of the
        parent pair.
    """

    def __init__(self, left=None, right=None, value=None):

        # The value of the current pair.
        self.value = value

        # The parent of the current pair.
        self.parent = None

        # The left element of the pair.
        self.left = left
```

```
        # The right element of the pair.
        self.right = right

    def create_tree(self, ids):
        """
            The Recursion converts the received ids list to a tree.
        """

        if len(ids) > 2:

            self.left = Pair()
            self.right = Pair()

            max_power = Pair.get_max_power(2, len(ids))

            ids_buffer = Pair.get_by_half(max_power, len(ids))

            if ids_buffer > len(ids) / 2.0:

                ids_buffer = len(ids) - ids_buffer

            if ids_buffer <= 1:
                self.left.create_tree([ids[0]])
            else:
                self.left.create_tree(ids[: ids_buffer])

            if len(ids) - ids_buffer <= 1:
```

```
            self.right.create_tree([ids[-1]])
        else:
            self.right.create_tree(ids[ids_buffer:])

    elif len(ids) == 2:

        self.left = Pair(value=ids[0])
        self.right = Pair(value=ids[1])

    elif len(ids) == 1:

        self.left = Pair(value=ids[0])
        self.right = None

def vlr(self, monitors_dict=None, recorder=None):
    """
            A recursion on the current tree of the form: value, left, right.
            This recursion creates the splitters and the monitors in the
            tree.
    """

    # It's not a monitor, it's a splitter.
    if not self.value:

        # In case it's the root of the tree set the parent panel of the splitter to
        # the parent of the root. It doesn't have value attribute like a pair
        # instance.
        if not isinstance(self.parent, Pair):

            # True if there is only one monitor in the whole tree.
```

```
        if not self.right:

            # Extract the id of the monitor in the left element of the pair.
            monitor_id = self.left.value

            # Convert this pair to a monitor.
            self.value = Monitor(self.parent, monitor_id)

            # Both its elements should be None.
            self.left = None

            # Update the monitor as described in the function.
            self.value.update_monitor(recorder, monitors_dict)

        else:

            # Create the splitter.
                self.value = wx.SplitterWindow(self.parent,
                style=wx.SP_LIVE_UPDATE)

            # Set the minimum size of the splitter's panels so the wouldn't
            # be too small.
            self.value.SetMinimumPaneSize(50)

    # It's not the first pair in the tree.
    else:

        # This pair instance has only one monitor. Therefore it shouldn't
            be a splitter. Convert this pair to
```

```
        # its left pair object.
        if not self.right:

            # Convert this pair to a monitor.
            self.value = Monitor(self.parent.value, self.left.value)

            # Both its elements should be None.
            self.left = None

            # Update the monitor as described in the function.
            self.value.update_monitor(recorder, monitors_dict)

        # It's a regular splitter.
        else:

            # Create the splitter.
            self.value = wx.SplitterWindow(self.parent.value,
            style=wx.SP_LIVE_UPDATE)

            # Set the minimum size of the splitter's panels so the wouldn't
            # be too small.
            self.value.SetMinimumPaneSize(50)

        # Create a sizer.
        sizer = wx.BoxSizer(wx.VERTICAL)

        # Add the splitter / monitor to the sizer.
        sizer.Add(self.value, 1, wx.EXPAND)
```

```
        # Set the sizer as the parent's sizer.
        self.parent.value.SetSizer(sizer)

    # Keep iterating on the left pair of the current pair.
    if self.left:

        self.left.parent = self
        self.left.vlr(monitors_dict, recorder)

    # Keep iterating on the right pair of the current pair.
    if self.right:

        self.right.parent = self
        self.right.vlr(monitors_dict, recorder)

# It's a monitor (of a splitter), the lowest pair in the tree.
else:

    # Create the monitor.
    self.value = Monitor(self.parent.value, self.value)

    # Create a sizer.
    sizer = wx.BoxSizer(wx.VERTICAL)

    # Add the monitor to the sizer.
    sizer.Add(self.value, 1, wx.EXPAND)
```

```
        # Set the sizer as the parent's sizer.
        self.parent.value.SetSizer(sizer)

        # Update the monitor as described in the function.
        self.value.update_monitor(recorder, monitors_dict)

    def lrv(self, flag):
        """
                A recursion on the current tree of the form: left, right, value.
                This recursion split the splitters. The

                flag parameter is for determining weather there is a need for a
                horizontal split or a vertical one.
        """

        if self.left:

            self.left.lrv(not flag)

        if self.right:

            self.right.lrv(not flag)

        # Split only if there are two elements for the current pair instance.
        if self.left and self.right:

            if flag:

                self.value.SplitHorizontally(self.left.value, self.right.value)
```

```
        else:

            self.value.SplitVertically(self.left.value, self.right.value)

    @staticmethod
    def get_max_power(base, number):
        """
                This function receives a base of a power and a number. The
                function the calculates what is the highest power

                that its result is less than the received number. This function
                doesn't handle usage of number < 1, i.e.

                the smallest base that can be returned is 0.
        """

        if (number**0.5 - floor(number**0.5)) == 0 or number == 2:
            return number

        current_power = 1

        while True:

            if base**current_power > number:

                return base**(current_power - 1)

            else:

                current_power += 1
```

```
@staticmethod
def get_by_half(max_power, number):
    """
        The function receives a power and its max power by base 2.
        Returns the right ids_buffer value.
    """

    if max_power*1.5 > number:

        return max_power / 2

    return max_power
```

הוצאה לפועל של הקובץ להרצה

```
# The loop which runs the file to run.
    while not self.general.shut_down or not self.status == "Finished":

        # Receive the output of the file by lines.
        line = self.process.stdout.readline()

        # The code have finished running.
        if line == '' and self.process.poll() is not None:

            # Make sure that the process has finished writing everything to
the log file.
            self.process.wait()

            # The rest of the output.
            output = self.process.stdout.read()
```

```
            # Write to the log file the rest of the content.
            with open(self.log_file, "a") as log_file:
                log_file.write(output)

            # Send the server the whole output of the running file since the
beginning.
            with open(self.log_file, "rb") as log_file:
                output = log_file.read()

self.general.thread_com.send.current_messages.append("Request::" +
str(self.request.id) +
                                                    "::Last Output::" + output)

            break

        # Attach the new line to the output list ans send it to the server.
        # Do it only if it's not a blank line.
        if line and line != "\n":

            self.stdout.append(line)

            with open(self.log_file, "a") as log_file:

                log_file.write(line)

self.general.thread_com.send.current_messages.append("Request::" +
str(self.request.id) +
                                                    "::New Output::" + line)
```

שליחה של בקשה במערכת

```
  def send_request(self, socket, dialog_box, general, header,
request_id=""):
    """
        The function sends the request to the socket received and prints to
the received dialog box that the request
        has received properly. The general argument is for notifying that
the request has sent.
    """

    self.prepare_to_sending()

    # Convert the request of the user to dictionary.
    request_dict = RequestFileCom.file_to_dict(path="Requests/" +
"".join(self.file_to_run.split(".")[:-1]) +
                                                "/current_request.txt")

    # The first thing to send is the file to run.
    file_to_run = request_dict["Run File"][:-1]
    header += "::Request::"

    if request_id:
        header += request_id + "::"

    # Send the server that the current client wishes to send him a new
request. In addition, send him the name of
    # The request.
    socket.send(header + "New Request::" + self.file_to_run)

    bytes_to_read = 1024 - len(header + "Uploading::Run File::")
```

```
    # Send the socket the file to run.
    with open(file_to_run, "rb") as f:

        content = f.read(bytes_to_read)
        content.replace("::", "%~")

        while len(content) >= bytes_to_read:

            socket.send(header + "Uploading::Run File::" + content)

            content = f.read(bytes_to_read)
            content = content.replace("::", "%~")

        if content:

            socket.send(header + "Uploading::Run File::" + content +
"::~~Finished Sending The File~~")

    # Send the dictionary of the current request file.
    socket.send(header + "Uploading::Request Dict::" +
json.dumps(request_dict))

    # Send the Commitments and Privileges values.
    socket.send(header + "Uploading::Executors Amount::" +
RequestFileCom.get_value(
        "Executors Amount", path="Requests/" +
"".join(self.file_to_run.split(".")[:-1]) +
                        "/current_request.txt")[:-1])
```

```
    if "Additional Files" in request_dict:
        num_of_additional_files = len(request_dict["Additional
Files"].split("@**"))
    else:
        num_of_additional_files = 0

    while not self.target_received_full_request:

        # Iterate over all the titles in the dictionary and send them with their
values to the server.
        for title in request_dict:

            # True If the file to run has sent and there are more additional file
that has to be sent to the server.
            if self.received_file_to_run and title == "Additional Files" and
num_of_additional_files >= self.\
                sent_additional_files_index + 1:

                # If the current file hasn't been sent
                if not self.sent_current_additional_file:

                    # All the additional file paths.
                    additional_files_paths = request_dict["Additional Files"]

                    # Iterate over all the additional file paths and send each
additional file to the server.
                    additional_file_path =
additional_files_paths.split("@**")[self.sent_additional_files_index]

                    if additional_file_path[-1] == "\n":
```

```
                additional_file_path = additional_file_path[:-1]

            additional_file_name = additional_file_path.split("\\")[-1]

            bytes_to_read = 1024 - len(header + "Uploading::Additional
File::" +
                                additional_file_name + "::")

            # Send the server the additional file.
            with open(additional_file_path, "rb") as f:

                content = f.read(bytes_to_read)
                content.replace("::", "%~")

                while len(content) >= bytes_to_read:

                    socket.send(header + "Uploading::Additional File::" +
                            additional_file_name + "::" + content)

                    content = f.read(bytes_to_read)
                    content = content.replace("::", "%~")

                if content:

                    socket.send(header + "Uploading::Additional File::" +
additional_file_name
                            + "::" + content + "::~~Finished Sending The
File~~")
```

```
                self.sent_current_additional_file = True

            # The server has received the current additional file.
            elif self.server_received_current_additional_file:

                self.sent_additional_files_index += 1
                self.sent_current_additional_file = False
                self.server_received_current_additional_file = False

        # True if the target has received the whole request properly.
        if self.received_file_to_run and num_of_additional_files <=
self.sent_additional_files_index:

            self.received_file_to_run = False
            self.sent_additional_files_index = 0
            self.sent_current_additional_file = False
            self.server_received_current_additional_file = False
            self.target_received_full_request = True
            if dialog_box:
                dialog_box.update_text("Request has received properly.")

            socket.send(header + "Finished Sending")

    general.send_request = False
```

פרק ו'- בדיקות

שם הבדיקה	מטרת הבדיקה	מה נדרש לבצע	מתי	מה בוצע בפועל
רגיל	לבדוק האם המערכת מתפקדת כראוי ללא מקרי קיצון.	להריץ לפחות חמישה מוציאים לפועל ושלושה משתמשים. לשלוח 3 בקשות שירוצו במקביל המבקשות טיפול מכל המוציאים לפועל.	28.4.17	תכנת המפעיל הופעלה כהלכה. 8 מוציאים לפועל התחברו למערכת כהלכה. 3 משתמשים התחברו כהלכה. כל אחד מהמשתמשים שלח בקשה לשרת. השרת הריץ את שלושת הבקשות במקביל כך שכל בקשה רצה על כל שמונת המוציאים לפועל.
ניתוק סטטי	לבדוק האם המערכת יודעת להתמודד עם ניתוקים של ישויות המחוברות אליה כשהיא במצב סטטי.	לנסות לנתק כל אחת מהישויות המחוברות למערכת לאחר הרצה של מספר בקשות.	28.4.17	חזרה על מה שבוצע בבדיקה **רגיל** ובסיומה ניתוק שלושה מוציאים לפועל ומשתמש. המפעיל התעדכן בניתוק והמערכת המשיכה לפעול כרגיל.
ניתוק דינמי	לבדוק האם המערכת יודעת להתמודד עם	לנסות לנתק כל אחת מהישויות תוך	28.4.17	חזרה על מה שבוצע בבדיקה **רגיל** רק שלפני שהבקשות הסתיימו לרוץ בוצע ניתוק של שני

שם הבדיקה	מטרת הבדיקה	מה נדרש לבצע	מתי	מה בוצע בפועל
	ניתוקים של ישויות המחוברות אליה כשהיא במצב דינמי.	כדי הרצה של בקשות.		מוציאים לפועל. המוניטורים שלהם במסכים של כל אחד מהמשתמשים הפסיקו להתעדכן. לאחר מכן נותק משתמש (תוך כדי שהמוציאים לפועל הריצו את הבקשה שלו), המוציאים לפועל הפסיקו להריץ את הבקשה שלו והשרת עדכן את המפעיל.
בדיקת הקלט של הלקוח	לבדוק שהלקוח לא מכניס בקשה שהשרת לא יכול להתמודד אתה.	להכניס קלטי קיצון ולנסות לחבל בבקשה שתשלח לשרת.	28.4.17	שונה התוכן של הקבצים המצורפים, התוכן התעדכן כהלכה. צירוף של קבצים עם תווים שלא נתמכים או סוג לא נתמך ושינוי path שלהם הביאו להסרתם. בקשה של יותר מוציאים לפועל משהשרת יכול לספק הביא לאי הרצת הבקשה. כל קלט שאינו תקין לווה בהודעה ללקוח.

פרק ז'- מדריך למשתמש

למשתמש

מטרת התוכנה היא לאפשר לך להריץ קוד על מספר מחשבים בכיתה מבלי שתצטרך לקום אליהם ולעשות זאת בעצמך. בחלק העליון של החלון ניתן להבחין בשתי כרטיסיות: Request, Monitor. בכרטיסיה Request תוכל להרכיב בקשה. הבקשה למעשה אומרת למחשב הראשי במעבדה איך אתה רוצה שהקוד שלך ירוץ. הבקשה תשלח למחשב הראשי ברגע שתלחץ על הכפתור "Send Request" בפינה השמאלית העליונה של החלון.

כעת נעבור לאופן **הרכבת הבקשה**:

צירוף קוד להרצה: הקוד הינו המרכיב החשוב ביותר בבקשה ועל כן בקשה לא יכולה להתממש אם לא תצרף אותו. כדי לצרף קוד לבקשה עליך לגרור את הקובץ המכיל אותו ולשחרר אותו מעל התיבה תחת הכותרת "File to run". **שים לב**, ישנם מקרים שבהם הקובץ לא יתקבל כהלכה ולכך כמה סיבות:

- הקוד מכיל תווים שהמערכת לא תומכת בהם כמו: ":".
- התוכנה מאיזושהי סיבה לא מצליחה לקרוא אותו.
- הכתובת שלו שונתה.
- סוג הקובץ לא נתמך.

הערות נוספות:

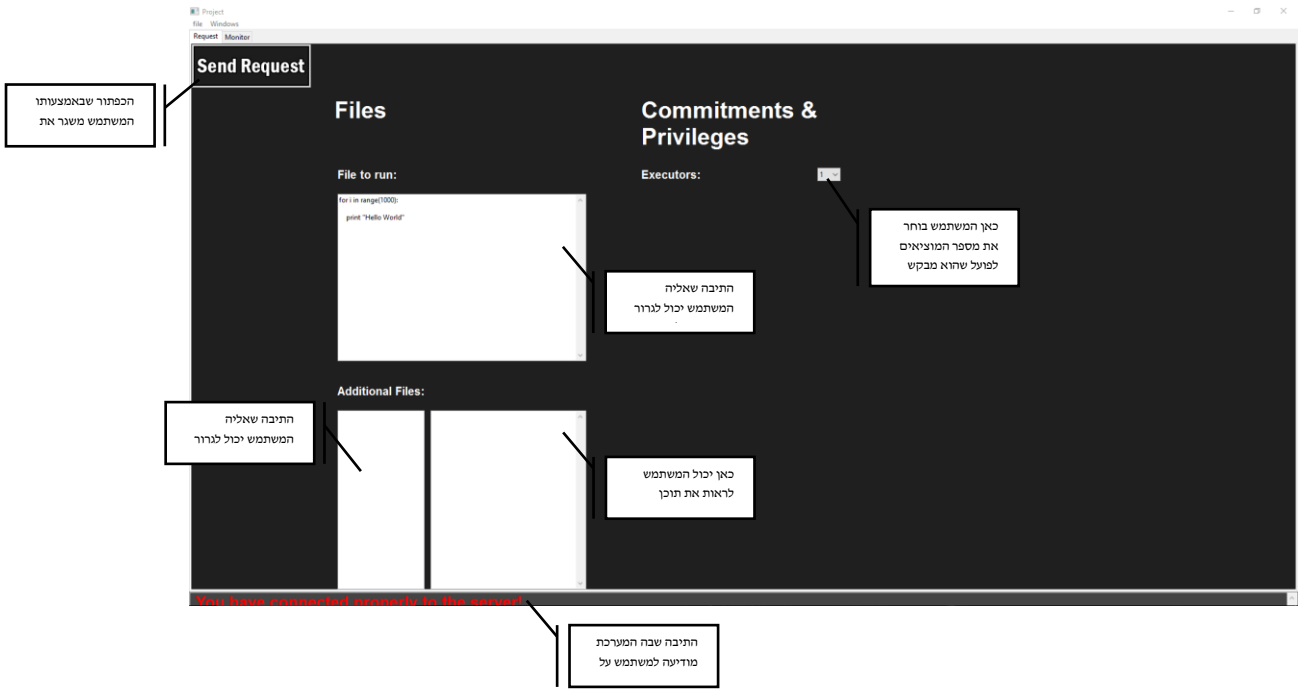
- ניתן להחליף קובץ בכל זמן שהוא על ידי גרירה ושחרור של קובץ אחר.
- ניתן לשנות את תוכן הקובץ. התוכן אמור להתעדכן בחלון אך גם אם לא התוכנה משתמשת בקובץ עצמו ולא בתוכן שבתיבה.
- אם התוכנה תזהה את אחת הבעיות לעיל תוכן הקובץ ימחק מהתיבה ותוצג הודעה מתאימה בתיבת ההתראות שבתחתית החלון.

צירוף קבצים נלווים: הבקשה לא חייבת להכיל קבצים נלווים. הקבצים הנלווים ישמרו באותה תיקייה שממנה ירוץ הקובץ להרצה. כדי לצרף קבצים נלווים (אפשר לצרף אותם אחד אחד ואפשר לצרף כמה בפעם אחת) עליך לגרור אותם ולשחרר בתיבה השמאלית תחת הכותרת "Additional Files". אם ברצונך להציג את התוכן

של קובץ נלווה פשוט לחץ על שמו בתיבה השמאלית והוא יופיע בתיבה הימנית. התוכנה לא תציג תוכן של קבצים מסוגים מסוימים, כגון קבצי תמונות למיניהם.

<u>מספר המוציאים לפועל:</u> ביכולתך לקבוע כמה מוציאים לפועל אתה דורש שיממשו את הבקשה. יתכן שתבקש יותר מוציאים לפועל משהמערכת יכולה לספק לך באותו רגע נתון. במקרה כזה הבקשה לא תצא לפועל ותופיע הודעה חדשה בתיבת ההתראות שבה יהיה כתוב מספר המוציאים לפועל המקסימלי שאתה יכול לבקש עם הבקשה הנוכחית באותו רגע נתון.

כך נראית הכרטיסייה Request:



אם לא קיבלת התראה בנוגע למשהו לא תקין בבקשה לאחר שלחצת על הכפתור "Send Request", הבקשה כרגע מורצת במערכת על ידי המוציאים לפועל.

שים לב, במקרים של עומס על המערכת; בקשה שמכילה קבצים כבדים, מספר המוציאים לפועל המבוקש גדול, מחשבים לא חזקים ועוד, עלול לקחת זמן עד שיתקבל מידע על הבקשה ולכאורה לא יקרה כלום. אנא המתן בסבלנות.

כשהבקשה תסתיים תקבל הודעה מתאימה בתיבת ההתראות.

כעת נעבור לראות **כיצד ניתן לעקוב אחר הבקשה**:

ראשית עליך לעבור לכרטיסיה Monitor. בכרטיסיה זו יוצגו לפניך פאנלים כמספר המוציאים לפועל שיצרו קשר עם המערכת בנוגע לקובץ הרץ. ניתן להגדיל ולהקטין את הפאנלים בעזרת גרירת הקווים המפרידים ביניהם. בכל אחד מהפאנלים תוכל לראות:

- את הid של המוציא לפועל.
- את הפלט של הקובץ הרץ ואת הודעות השגיאה אם יש כאלה.

כך נראית הכרטיסייה Monitor:

למפעיל

כמפעיל המערכת אתה יכול לעקוב אחר המצב הנוכחי של המערכת ולהיחשף לתיעוד שנשמר במסד הנתונים מפעילויות ישנות יותר. כמו כן יש לך היכולת לשלוט על מצבה הנוכחי.

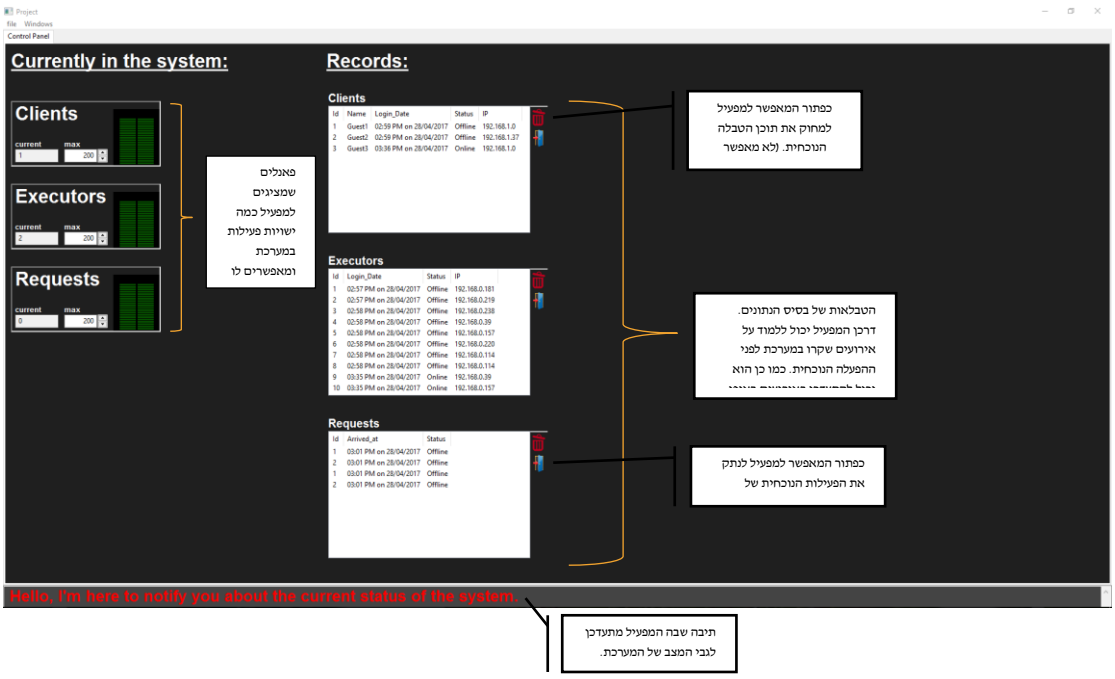
תחת הכותרת "Currently in the system" תוכל למצוא שלושה פאנלים כשכל אחד מהם מציג את מספר הישויות שהוא מייצג שכרגע פעילות במערכת. בתיבה תחת הכותרת "current" נמצא מספר הישויות הפעילות נכון לרגע הנוכחי. באפשרותך להגביל את מספר הישויות על ידי שינוי המספר שנמצא תחת הכותרת "max".

תחת הכותרת "Records" תוכל לראות את הטבלאות שנמצאות בבסיס הנתונים ואת תוכנן. מימין לכל טבלה תוכל למצוא סרגל המכיל כפתורים שמאפשרים לך לבצע פעולות על הטבלה שמשמאלם.

מחיקת תיעוד שאינו פעיל: אם ברצונך להיפטר מהתיעוד בטבלה אתה יכול ללחוץ על כפתור פח האשפה. לאחר הלחיצה כל השורות המייצגות דברים שלא פעילים יותר תמחקנה.

הפסקת פעילות: אם ברצונך לנתק לקוח או מוציא לפועל, או להפסיק פעילות של בקשה מסוימת, תוכל לסמן את השורה שלהם (אפשר כמה בפעם אחת) וללחוץ על כפתור היציאה (דלת פתוחה עם חץ).

כך נראה חלון הבקרה של המפעיל:



פרק ח'- מבט אישי

מבט אישי על העבודה ועל תהליך פיתוחה

אתגרים

אין ספק שהפרויקט הציב בפני אתגרים רבים. אחד האתגרים המשמעותיים ביותר היה ההתמודדות עם המודול הגרפי החדש שהשתמשתי בו בפיתוח התוכנה. גיליתי שwxpython מעט מורכב יותר משחשבתי בהתחלה. בכדי להשתמש בו בצורה הנכונה יש לעשות שימוש בדברים לא אינטואיטיביים כל כך, אם לא פועלים לפי ההיררכיה שהוא מושתת עליה נאלצים לפעמים להתחיל דברים מההתחלה.
כשהתחלתי לעבוד אתו הבנתי שקפצתי למים העמוקים. לכן החלטתי שאני משקיע בין יום ליומיים רק בלמידה על המודול. רק לאחר שהבנתי את הרעיונות המרכזיים לעומק והכרתי עקרונות חיוניים הצלחתי לעבוד עם המודול בקלות.

אתגר נוסף היה הכנת המוניטור למשתמש. ישנם דרכים פשוטות למימוש מוניטור כזה אך הן לא גרמו לי להתפשר. דרך אחת הייתה להוסיף בכל פעם מוניטור נוסף אל תחתית העמוד והמשתמש יגלול למעלה ולמטה בכדי לדלג בין המוניטורים. דרך אחרת הייתה להציג מספר מצומצם של מוניטורים, אפילו אלו שהמשתמש יבחר. אך אני חיפשתי פתרון שיאפשר למשתמש לראות את כל המוניטורים בבת אחת ובהיררכיה מסוימת. זהו אינו פתרון של מה בכך כמו שאר הפתרונות ולקח לי שבוע להביא אותו לשלמות. את הפתרון ניתן למצוא בפרק ה'.

הערכת הפתרון לעומת התכנון והמלצות לשיפורו

התוצר הסופי של המערכת שונה מאוד מהתוצר שתכננתי לפני הפיתוח. אני מעריך שהשינויים שביצעתי לא היו רק מקצועיים אלא גם נבעו מתוך שיקולים אחרים כמו עמידה בלוח זמנים וסדר עדיפויות. עם זאת, אני מרוצה מהפתרון הסופי משום שהוא ממלא את המטרה שלשמה יצרתי אותו באופן מספק למדי. ישנם המון שיפורים שאפשר להוסיף לו, ביניהם:

- הוספת אלגוריתם בשרת שימיין את הבקשות בין המוציאים לפועל באופן היעיל ביותר.

- שדרוג ממשק המשתמש כך שיספק לו שליטה על כל מוציא לפועל בנפרד ויאפשר לו לבצע פעולות נוספות.
- תמיכה בתכונות נוספות שיגדילו את טווח האפשרויות להרצה, כגון: תמיכה בסוגים נוספים של קבצי הרצה, תמיכה בקלט ועוד.
- לפתרון הסופי חסרה שכבת אבטחה נוספת שתמנע פעולות זדוניות מתקדמות.

פרק ט'- ביבליוגרפיה

רקע תיאורטי

במהלך פיתוח ותכנון הפרויקט נאלצתי ללמוד דברים חדשים שלא שלטתי בהם טוב מספיק ולחפש פתרונות לבעיות שנתקלתי בהן. עוד בשלב התכנון, כשרציתי לבסס את המערכת על מכונות וירטואליות, הייתי צריך להעמיק בתחום. בנוסף, עשיתי שימוש בבסיס נתונים ולכן הייתי צריך לרענן את הידע שלי בsql ובבסיסי נתונים בכלל. כמו כן למדתי רבות ממקורות שונים על המודול הגרפי שהשתמשתי בו להכנת הguי למשתמש ולמפעיל. במשך פיתוח הקוד נתקלתי בלא מעט מקרים שחשדתי שקיים פתרון טוב יותר מזה שמצאתי או שהייתי צריך פתרון שלא היו לי האמצעים לפתח אותו בכוחות עצמי ולכן פניתי לפורומים שונים באינטרנט.

ספרות מקצועית

Wxpython

- The Mouse VS. The Python : https://www.blog.pythonlibrary.org/
- ZetCode : http://zetcode.com/
- https://wxpython.org/docs/api/wx-module.html
- Tutorialspoint : https://www.tutorialspoint.com/index.htm
- DZone : https://dzone.com/articles/wxpython-wxlistctrl-tips-and

Virtual-Machines

- Oracle VirtualBox : https://www.virtualbox.org/
- VMWare : http://www.vmware.com/

SQL

- Khan Academy : https://www.khanacademy.org/computing/computer-programming/sql/sql-basics/v/welcome-to-sql

שאלות ותשובות

- http://stackoverflow.com/questions/42495628/transfer-files-to-vm
- http://stackoverflow.com/questions/43088032/popen-stdout-log-file-and-pipe
- http://stackoverflow.com/questions/41872675/load-balancing-server-clients