

046273 - תכנות פונקציונלי ומבוזר

תרגיל בית 3: תכנות מקבילי מתקדם

תאריך הגשה מפורסם ב-moodle

תרגיל ראשון – שרת מבוזר ומקבילי מבוסס GenServer לחישוב פונקציות

בתרגיל זה תממשו מערכת המכילה שלושה שרתים המבצעת חלוקת עומסים. כל שרת ידע לקבל הודעות המכילות פונק' לביצוע. השרתים ידאגו כך שמספר הפונק' שמורצות ע"י כל שרת יהיה זהה (ככל הניתן) עבור שלושת השרתים השונים.

השרתים במערכת ימומשו באמצעות GenServer. השרתים ירוצו על Node משותף (לצורך פשטות). כל השרתים יוגנו ע"י supervisor משותף.

השרתים ירשמו בשמות **server1, server2, server3**.

שימו לב – כל אחד מהשרתים יכול להריץ מספר רב של פונק' במקביל. כלומר, הרצת הפונק' עצמה מתבצעת בתהליך אחר מתהליך השרת, כך שכל הודעה המגיעה לשרת זוכה לטיפול מיידי. רמז – `handle_cast`. רמז 2 – בכל מבחני העבר בקורס הייתה שאלה איך ממשים `gen_server` מקבילי. מוזמנים לחשוב מראש וליישם בתרגיל.

על המימוש לכלול מודול בשם **loadBalance** המכיל ועושה `export` לפונקציות הבאות:

`startServers/0`

תפעיל את המערכת כולה.

`stopServers/0`

תעצור את המערכת כולה.

`numberOfRunningFunctions/1`

הפונק' תקבל כפרמטר את **מספר השרת**. הפונק' תחזיר את מספר הפונק' הרצות שהשרת המבוקש אחראי עליהן.

לדוגמא – אם שרת מספר 1 אחראי על הרצה של 4 פונק' ושרת מספר 2 אחראי על הרצה של 3 פונק', הרי שנקבל:

`numberOfRunningFunctions(1) -> 4`

`numberOfRunningFunctions(2) -> 3`

calcFun/3

הפונק' תקבל כפרמטר ראשון **PID** של לקוח, כפרמטר שני **פונקציה F כלשהי שאינה מקבלת פרמטרים** וכפרמטר שלישי **MsgRef**. הפונק' תחזיר את האטום ok עבור כל הרצה (שימו לב - בלי סוגריים). כמו כן, הפונק' תוסיף את F לשרת שלו המספר המינימלי של פונק' רצות. לאחר ההוספה, מספר הפונק' הרצות של שרת זה יגדל ב-1. בעת סיום הרצת הפונק' F , הערך המוחזר שלה וה-MsgRef ישלח כהודעה ל-PID. מבנה ההודעה – {MsgRef,F_result}.

לדוגמא – נניח ששרתים 1,2 אחראים כרגע על הרצת 5 פונק' כל אחד ושרת 3 אחראי על 4 פונק'. לאחר הרצת calcFun(self(),fun() -> timer:sleep(3),5 * 5 end,MsgRef) כל שלושת השרתים יהיו אחראים על 5 פונק' כל אחד. לכן הפעלת numberOfRunningFunctions עם פרמטר 1/2/3 יחזיר את הערך 5. ערך ההחזרה של הקריאה עצמה ל-calcFun יהיה כאמור ok. בהמשך הדוגמא, לאחר 4 שניות למשל, תגיע ההודעה {MsgRef,25} חזרה לתהליך שקרא לפונק' calcFun. לכן, הפעלת numberOfRunningFunctions עם פרמטר 3 יחזיר כעת את הערך 4.

סה"כ 4 פונקציות למימוש.

ניתן לייצר ולהשתמש במודולים נוספים כראות עיניכם, אך כמובן יש לצרפם להגשה.

לסיכום – מערכת של שלושה שרתים מבוססי GenServer **מקביליים** ומוגנים באמצעות Supervisor אשר אחראים על הרצת פונק' המתקבלות כקלט. כל שרת יידע לספר בכל רגע נתון על כמה פונק' הוא אחראי. כאשר פונק' מסיימת לרוץ, ערך ההחזרה שלה נשלח כהודעה ל-PID שניתן בעת שליחת ההודעה לשרת.

ניתן לכם חופש לתכנן את המערכת כרצונכם, אך חייבת להיות תמיכה בארבעת הפונק' שצויינו.

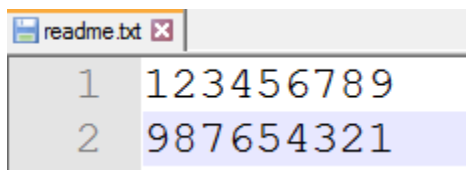
הוראות הגשה:

- הגשה בזוגות בלבד. ניתן להיעזר בפורום באתר הקורס ע"מ למצוא שותף.
- הגשה אלקטרונית דרך מערכת ה-Moodle. הגשה אחת בלבד עבור שני בני הזוג. שאלות על תוכן התרגיל – דרך הפורום המתאים ב-Moodle.
- עליכם להגיש את כל קבצי המקור (כלומר, .erl) הכוללים תיעוד ומימוש. אין להגיש קבצים מקומפלים (.beam) או תתי-תיקיות.
- לא ניתן להגיש את התרגיל באיחור למעט מקרים פרטניים (מילואים וכו') – בתיאום **מראש** עם המרצה.

בדיקת התרגילים מתבצעת באופן אוטומטי, ולכן עליכם לוודא לפני ההגשה:

- שמות המודולים, הפונקציות והקבועים תואמים לשמות המצוינים בהנחיות התרגיל.
- לא מתבצעות הדפסות לא רצויות למסך.
- הקוד מתקמפל ורץ כהלכה.
- טיפול במקרי קצה – מספרים שליליים ואפס, רשימות ריקות וכו'. מדיניות הטיפול בשגיאות הינה זו שהוגדרה במהלך הקורס - let it crash.

צרו קובץ בשם readme.txt המכיל את ת"ז של שני המגישים בשתי שורות נפרדות. לדוגמא:



```
1 123456789
2 987654321
```

את כל הקבצים יש לארוז בקובץ ZIP (בלבד). שם קובץ ההגשה יהיה

ERLANG_HW3_<id1>_<id2>.zip

כאשר id1 ו-id2 הינם מספרי תעודות הזהות של המגישים. לדוגמא:

ERLANG_HW3_123456789_987654321.zip

לאחר הגשת הקבצים ב-Moodle, אתם נדרשים להוריד את הקבצים שהגשתם ולראות כי הם מתקמפלים ורצים בסביבה נקייה.

הבהרה – אי עמידה בהוראות ההגשה שקולה לאי הגשת התרגיל. אתם נדרשים לוודא כי קיימתם את כל ההוראות המפורטות. לא תהיינה הגשות חוזרות.

בהצלחה!