רבעון שני 2024 גיליון מס' 37

מגזין עולם הבדיקות

www.testingworld.co.il

בדיקות ביצועים ועומסים עם JMETER או K6 בעידן AI/ML

שי גינזבורג

החשיבות בבדיקות זיכרון במכשירי אנדרואיד

טופז אנגלנדר

בחירת שפת תכנות עבור תשתית אוטומציה

ניסים אריאל

בחן את עצמך

ירון צוברי

דבר העורך
ניצן גולדנברג

ניצן גולדנברג

מזה 8 שנים בתחום בדיקות התוכנה, תפקיד נוכחי מהנדס בדיקות בכיר בחברת SeatGeek, מנהל קבוצת ה-AB של ארגון ITCB® המוביל הראשי של קבוצת המיטאפ TestIL, מרצה בקורסים לבודקי תוכנה

קוראים יקרים,

מונח לפניכם גליון מס 37 של מגזין "עולם הבדיקות".

בשל המצב הבטחוני בישראל, חלק מהכותבים שלנו במגזין נקראים למילואים ולכן הגליון יוצא בעיכוב קל, עמכם הסליחה.

בגליון זה תוכלו למצוא מגוון רחב של מאמרים חדשים, בנוסף לטורים המעולים והקבועים שלנו:

"בחירת שפת תכנות עבור תשתית אוטומציה" – ניסים סער אריאל

טופז אנגלנדר מסביר לנו על החשיבות בבדיקות זיכרון במכשירי האנדרואיד

שי גינזבורד חוזר בעוד מאמר מעניין על "בדיקות ביצועים ועומסים עם JMETER או K6 בעידן AI/ML"

יש לנו תשבץ וחידות חדשות ומאתגרות עבורכם בגליון זה

בנוסף, יש לנו את הטורים הקבועים: מחפש צרות, בחן את עצמך, ראיון עם מנהל/ת בדיקות, האנציקלופדיה לבדיקות, עושים QA לקריירה, משולחנו של שביט, ונגיסה מ-TestIL.

אנו נשמח לקבל בקשות לנושאים חדשים ומעניינים למאמרים, צרו עימנו קשר במייל

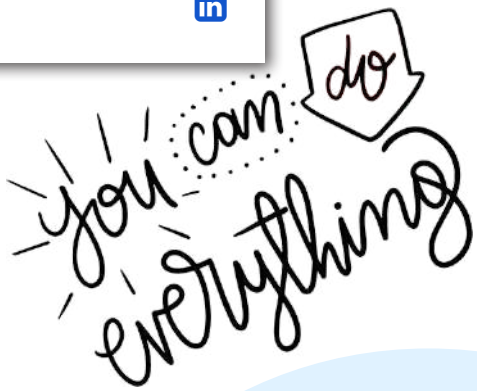
magazine@testingworld.co.il

ארכון מוסמכים בישראל

סה"כ מוסמכים בבורד הישראלי ITCB® – 10,309

מתוכם 966 "מצטיינים" אשר ענו מעל 90% תשובות נכונות.

80 מצטיינים ברבעון השני של שנת 2024



קריאה מהנה,
ניצן גולדנברג

מו"ל
Israeli Testing Certification Board
ITCB®

ניהול המגזין
יאן ברון, *iMDsoft*

ניהול התוכן
קובי הלפרין, *Red Hat*

עורך ראשי
ניצן גולדנברג, SeatGeek

עיצוב גרפי
בית נלי מדיה
סטניסלב קולנקו
www.beitnelly.com

יצירת קשר
אימייל:
magazine@testingworld.co.il

הרשמה
http://bit.ly/TW-Reg
פקס:03-6176605
כתובת: ברוך הירש 14 בני ברק 51202

www.testingworld.co.il

www.itcb.org.il

תוכן העניינים

דבר העורך 2

ראיון עם מנהל בדיקות | אור אמויאל CYREBRO 4

תחרות הבדיקות הישראלית ISTC 2024 7

בחן את עצמך - שאלה | ירון צוברי 8

החשיבות בבדיקות זיכרון במכשירי אנדרואיד
טופז אנגלנדר 10

משולחנו של שביט - מה עלול לקרות כשבודק אחד מחזיק בכל הידע? | שביט ג'רסי 12

עושים QA לקריירה - איך נראית השפעה שלא קשורה לטייטל שלך | איילת מלמד 13

בדיקת ביצועים ועומסים עם JMETER או K6 בעידן AI/ML | שי גינזבורג 15

האנציקלופדיה לבדיקות - בדיקות קומבינטוריות Combinatorial Testing | קובי יונסי 18

בחן את עצמך - תשובה | ירון צוברי 20

בחירת שפת תכנות עבור תשתית אוטומציה שיקולים, אילוצים ועוד | ניסים אריאל 21

מחפש צרות - עוד על פסיכולוגיה של בדיקות
מיכאל שטאל 23

נגיסה מ-TestIL מפגשים לבודקי תוכנה | ניצן גולדנברג 25

בודקים בכיף 26

דף העורכים 27

עולם הבדיקות נכתב ע"י בודקים עבור בודקים

ITCB® מקדמים את קהילת הבודקים בישראל

מגזין עולם הבדיקות

עולם הבדיקות הינו מגזין רבעוני.
כל הזכויות שמורות. זכויות היוצרים על חומר שפורסם על ידי המפרסם הינן רכושו של המחבר. הדעות המובאות במאמרים והתוכן לא בהכרח משקפים את דעת המפרסם. המחברים הינם האחראים הבלעדיים על תוכן מאמרם. מובהר כי העתקה ו/או נטילה שיטתית של מידע מהמגזין לצורך פעילות מסחרית ו/או עסקית, או לצורך כל פעילות אחרת שיש בה כדי לפגוע בפעילות העמותה, אסורה בהחלט.
לקבלת אישור לשימוש בתוכן צור קשר בדוא"ל **magazine@testingworld.co.il**.

ראיון עם מנהל בדיקות | אור אמויאל CYREBRO

אור אמויאל

שמי אור אמויאל, בן 28, מתגורר בחיפה עם בת זוגתי. ראש צוות בדיקות ואוטומציה בחברת CYREBRO, עליו הגעתי לאחר שקיבלתי הצעה מעניינת להיכנס לתחום הסייבר ולהשפיע באופן ישיר על תהליכי ביטחון המידע. צוותי מורכב מחמישה בודקים, שלושה מתמחים בבדיקות אוטומטיות ושניים בבדיקות ידניות. תחומי הפעילות שלנו כוללים בדיקות שרת ולקוח, ואנו מקדימים על דגש מיוחד בתהליכי CI/CD ובבדיקות מיקרו סרוויסים. כצוות מרכזי בחברה, אנו נותנים דגש רב להתאמה מושלמת של תהליכי הבדיקה לדרישות העסק והמשתמשים. לפני כל פעולה, אנו מתמקדים בהבנת המטרות והיעדים של החברה על מנת לוודא תרומתם האמיתית של הבדיקות. בנוסף, אנו חושבים בצורה מעמיקה על החלטות מבוססות נתונים בנוגע להשקעת המשאבים בפרויקטים השונים. אחד התחביבים הגדולים שלי הוא עזרה חברתית והתנדבות אם זה בבתי מחסה לבעלי חיים או בעזרה לזולת. אוהב לאתגר את עצמי ולעזור לבודקים מתחילים להתקבל למשרת ה-QA הראשונה שלהם.

מהן הדרישות הניהוליות והעסקיות מקבוצת הבדיקות שלכם נכון להיום ולעתיד?

כראש צוות האוטומציה והבדיקות במשך 3 השנים האחרונות, הדרישות הניהוליות והעסקיות מהצוות שלי הן מגוונות ומשתנות בהתאם לצרכים המתפתחים של הארגון. ברמה העסקית, אנחנו נדרשים לספק בדיקות איכות יסודיות ומקיפות לכל המוצרים והשירותים של החברה, תוך שמירה על לוחות זמנים קצרים ועמידה בתקציבים צנועים. במקביל, עלינו לתמוך בקצב הפיתוח המהיר ובשחרור גרסאות חדשות באופן תכוף.

מבחינה ניהולית, אני נדרש להוביל צוות בדיקות מקצועי, מסור ויצירתי, אשר יכול לעבוד בצורה יעילה תחת לחץ ולהתמודד עם אתגרים טכניים מורכבים. עלי לפתח תוכניות הדרכה ולטפח את הידע והמיומנויות של חברי הצוות באופן רציף.

בנוסף, עליי לקיים שיתוף פעולה הדוק עם צוותי הפיתוח והארכיטקטורה, כדי להבטיח אינטגרציה חלקה של תהליכי הבדיקות לאורך מחזור החיים של המוצר. גם נדרשת ממני תקשורת פנים-ארגונית טובה כדי לקבל החלטות על סדרי עדיפויות ולהציג דוחות ומצגות באופן ברור לדרגים הבכירים.

בעתיד הקרוב, אני צופה דרישה גוברת לאוטומציה של בדיקות, כולל פיתוח כלים ייעודיים המבוססים על בינה מלאכותית וללמידה חישובית, זה יחייב אותנו להיות גמישים ולהסתגל באופן מהיר.

מהן הדרישות הניהוליות והעסקיות מאנשי הבדיקות וממך כמנהל?

כמנהל צוות הבדיקות והאוטומציה , הדרישות הניהוליות והעסקיות ממני ומאנשי הצוות שלי הן מגוונות ומשמעותיות:

מאנשי הבדיקות נדרש:

1. ידע טכני מעמיק בתחומי הבדיקות, התשתיות והטכנולוגיות הרלוונטיות.
2. יכולת חשיבה אנליטית וביקורתית כדי לזהות חורים, סיכונים ובעיות אפשריות.
3. גמישות ויכולת להתאים את עצמם לשינויים תכופים בדרישות ובטכנולוגיות.
4. כישורי תיעוד וכתיבה טובים להכנת תוכניות בדיקות וניתוח תקלות.
5. עבודת צוות, תקשורת בין-אישית טובה ושיתוף פעולה עם צוותים אחרים.
6. יצירתיות בפיתוח כלים וסקריפטים אוטומציה חדשים.
7. נכונות למידה והתפתחות מקצועית רציפה.

מהיבט הניהולי, הדרישות ממני כראש הצוות כוללות:

1. מנהיגות טכנית וניהולית לצוות הבדיקות, קביעת תהליכים ויצירת תכנית עבודה ברורה.
2. ניהול המשאבים - כוח אדם, תקציבים ולוחות זמנים בצורה יעילה.
3. הערכה וטיפוח של הצוות, זיהוי צרכי הדרכה והכשרה.
4. תיאום ציפיות, סדרי עדיפויות וקביעת יעדים איכותיים ואסטרטגיים.
5. תקשורת בינארגונית עם דרגים בכירים, צוותי פיתוח ובעלי עניין אחרים.
6. הובלת שיפורים והתאמות בתהליכי העבודה וביישום טכנולוגיות חדשות.
7. ראיה כוללת של תמונת המצב והבנת הקשר בין הבדיקות לבין מטרות העסקיות.

איך נראה יום העבודה שלך?

כראש צוות בדיקות ואוטומציה בחברת CYREBRO, יום העבודה השגרתי שלי די עמוס ומגוון. בוקר טיפוסי מתחיל עם בדיקת המייל לשאלות/בעיות דחופות שזקוקות לטיפול שלי ועדכון על סטטוס שחרורים של לילה. לאחר מכן יש לי פגישת צוות קצרה עם צוות הבדיקות שלי כדי לסנכרן מול לוחות הזמנים, לקבוע סדרי עדיפויות לאותו יום ולתת משוב והדרכה במידת הצורך.

לאחר מכן אני בדרך כלל יוצא לפגישות שונות - עם צוותי הפיתוח והמוצר כדי לתכנן ולתאם תוכניות בדיקות לשחרורים הבאים, לקבל עדכונים טכניים וללמוד על דרישות חדשות. פגישות תיאום גם עם אדמינים, אנשי תשתיות, אבטחת מידע וכיוצא בזה. לעתים קרובות אני גם נדרש להציג מצגות סטטוס לדרגים בכירים במידת הצורך.

חלק ניכר מהיום שלי עובר בבחינה שוטפת של דוחות בדיקה, ניתוח כשלים קריטיים, פתרון בעיות בקרי שחרור והשקת סיבובי רגרסיה חדשים. אני גם משקיע זמן רב בתכנון וביצוע מהלכי שיפור בתהליכי הבדיקות ומנהל פרויקטי פיתוח כלים לאוטומציה חדשים.

לקראת הערב, יש לי לרוב עוד חלון של שעתיים-שלוש לעבודה שקטה על משימות יותר ארוכות טווח, כמו כתיבת תוכניות ומסמכי אסטרטגיה, לימוד טכנולוגיות חדשות, וביצוע ניתוחים מעמיקים של תוצאות בדיקה.

כמובן שבמקרים של אירועי חירום, כגון כשלי שחרור או פרצות אבטחה, סדר היום שלי עלול להשתנות לחלוטין.

יום העבודה שלי תובעני, אבל גם מגוון מבחינה טכנולוגית וניהולית, מה שהופך אותו למאתגר ומעניין. זה דורש יכולת הסתגלות מהירה, קבלת החלטות יעילה ומיקוד בעדיפויות משתנות.

מה האתגרים שלכם בתחום הבדיקות וכיצד אתם מתגברים עליהם, האם נראה כי אלו ישתנו בעתיד?

אנו מתמודדים עם מגוון אתגרים מורכבים, שחלקם צפויים גם להתעצם בעתיד. להלן כמה מהאתגרים המרכזיים ודרכי ההתמודדות שלנו:

1. קצב שחרורים מהיר ומורכבות טכנולוגית עולה - כדי לעמוד

ראיון עם מנהל בדיקות | אור אמויאל CYREBRO

בלחצים של מחזורי שחרור תדירים, הגדלנו משמעותית את היקפי האוטומציה של הבדיקות. אנחנו מיישמים כלים ומסגרות מתקדמות כמו Selenium, Locust ו- API Fortress, ומפתחים באופן עצמאי יכולות אוטומציה מותאמות אישית לצרכים הספציפיים שלנו.

2. פיצול בין פלטפורמות ומכשירים - עם הגידול בכמות המכשירים והסביבות, קשה להבטיח כיסוי בדיקות מספק. אנחנו נעזרים בשיטות כמו בדיקות רוחב לפלטפורמות מובילות ומרכזיות, שילוב למטריצת הבדיקות, ופיתוח מנגנוני אוטומציה המבצעים התאמה דינמית לפלטפורמות שונות.

3. דרישות רגולטוריות וציות - בעולמות כמו אבטחת מידע וסייבר, הציות לדרישות ולתקנים רגולטורים מחמיר. אנו מקפידים לתעד את תהליכי הבדיקות בצורה מפורטת וממוסדת, לאמת את כיסוי הבדיקות באופן שיטתי ודינמי ולשמור על ראיות ומסמכים ארוכי טווח.

4. תקציבים מוגבלים וקוצר זמן - עם זאת, מנהלים מצפים מאיתנו לספק כיסוי בדיקות מקיף עם משאבים מוגבלים. אנו מגבירים את היעילות באמצעות כלים לניהול תיקי בדיקות, מיקוד בסיכונים קריטיים, שיתופי פעולה עם צוותי פיתוח לבדיקות רציפות, ושכלול מתמיד של יכולות האוטומציה.

5. מחסור בכישורים נדרשים - כל הזמן נדרשים כישורים חדשים בתחום ה-DevOps, תשתיות ענן, למידה חישובית וכד'. אנו מעודדים למידה וחונכות פנים-צוותית, משתפים ידע עם צוותים אחרים, ויוצרים תוכניות הכשרה והסמכה מתמשכות.

לקראת העתיד, צפוי שאתגרים אלה יחריפו עם הגידול המשמעותי בנפחי הנתונים והמורכבות של היישומים. יהיה צורך להמשיך ולשכלל את יכולות האוטומציה על בסיס טכניקות כמו AI/ML, לאמץ פרדיגמות DevOps חדשניות ולהגביר את מעורבות קהילת הבדיקות בכל שלבי מחזור החיים. צוותי הבדיקות יצטרכו להיות יותר גמישים, יצירתיים ומקצועיים מתמיד.

באילו אתגרים נתקלים הבודקים שלך?

הבודקים שלנו נתקלים באתגרים טכניים מורכבים בפתרון בעיות בקוד ובהבנת הדרישות המורכבות של המערכת. הם עשויים להתמודד עם בעיות בהבנת התפקוד הצפוי של המערכת ובתהליכי הפיתוח.

בנוסף, צוות האוטומציה שלנו מתמקד בפתרון הבעיות בקוד של הפיתוח, מאפשר לנו להבטיח איכות ויציבות בעבודת הפיתוח ולהפחית את סיכוני התקלות טכניות במערכת.

באילו אתגרים ניהוליים הנך נתקל?

כמנהל, אני נתקל באתגרים ניהוליים כגון ניהול צוות גלובלי הנמצא באזורי זמן שונים אבל עדיין מחוייב לאיכות וזמני סיום. לצורך התמודדות עם אתגרים אלו, אנו מתמקדים בשיפור תהליכי העבודה ובשיפור היעילות של הצוות. זה כולל שיפור תהליכי הבדיקה והפיתוח, כגון הטמעת כלים וטכניקות חדשים, אוטומציה של תהליכי ניתוח ובדיקה, ושימוש במתודולוגיות כמו Agile ו-DevOps. מבחינתי כמנהל, אני נתקל באתגרים נוספים הקשורים לניהול צוות גדול ומגוון. זה כולל את הצורך לארגן תהליכי עבודה ולהקפיד על התאמה טובה של הצוותים השונים למטרות מגוונות, ואני מתמקד בשימוש בכלים ובשיטות שיעזרו לי לנהל את הצוות בצורה אפקטיבית ויעילה.

כיצג הנכם פועלים להעשרת הידע של הבודקים? (מתודולוגית וטכנית)

אני כולל בפעילות השוטפת שלי גם תהליך של העשרת הידע של הבודקים בצוות, כגון קידום אימון והכשרת הבודקים במתודולוגיות ובכלים חדשים, ודחיפה לשיתוף פעולה ולימוד משותף על מנת לשפר את יכולותיהם ולהגדיל את יעילותם. מבחינה ניהולית, אנו מתמקדים בפיתוח ובשיפור של יכולות הצוות, כולל שיפור הידע הטכני והמתודולוגי של הבודקים. אנו מקדישים זמן ומשאבים להכשרת הצוות בכלים וטכנולוגיות חדשות, כמו גם לשיפור היכולת שלהם להבין ולהתמודד עם בעיות טכניות מורכבות.

מה האתגרים שלכם בתחום הבדיקות?

התמודדות עם האתגרים בתחום הבדיקות דורשת הבנה מעמיקה של התהליכים והכלים הטכנולוגיים המתקדמים, ושימוש יציב במתודולוגיות כמו Agile ו-DevOps. בעתיד, אנו מצפים שהאתגרים בתחום יתמקדו יותר בצרכי האוטומציה והתפתחויות בפיתוח ובתהליכי הבדיקה הרצופים.

בתחום הסייבר, תהליכי הבדיקות משלבים בין בדיקות אוטומציה לבדיקות ידניות על מנת להבטיח איכות וביצועים מיטביים של המערכות המורכבות. הנה כמה מהכלים שאנחנו משתמשים בהם אצלנו בתוך הצוותים:

1. Selenium

2. Jenkins: מערכת CI/CD המאפשרת אוטומציה של תהליכי בניה ופרסום קוד.

3. Burp Suite: בדיקות אבטחה ופריצות באפליקציות רשת.

4. Postman: לבדיקות API

5. Locust: לבדיקות עומסים.

בנוסף לכלים אלו, תהליכי הבדיקות בעולם הסייבר דורשים ידע מעמיק בתהליכי פיתוח ובטכנולוגיות מתקדמות, ויכולת ליישם אותם באופן יעיל ויציב תוך שימוש במתודולוגיות רצופות כדי להבטיח איכות ואמינות ברמה גבוהה.

מה היית ממליץ לבודק תוכנה שנמצא בתחילת הקריירה שלו?

1. השקיעו בלימוד מתמיד - תחום בדיקות התוכנה משתנה במהירות עם כלים, טכנולוגיות ושיטות חדשות שצצות כל הזמן. הפכו את הלמידה להרגל קבוע - קראו בלוגים, צפו בקורסים ממוקדים, השתתפו בכנסים מקצועיים וחפשו הזדמנויות לצבור הסמכות ותעודות רלוונטיות. הכשירו את עצמכם במיוחד בכלים פופולריים כמו Selenium, Appium, Postman ובמתודולוגיות אג'יל ותהליכי DevOps.

2. פתחו חשיבה אנליטית וביקורתית - כבודקי תוכנה, עלינו לפתח כישורי חשיבה לוגית ויכולת זיהוי וניתוח דפוסים, חריגים ותרחישי כשל אפשריים. התאמנו בלמצוא דרכים לבחון את הדברים מזוויות שונות וחשבו על מקרים קיצוניים.

3. התמחו בתחום ספציפי - יתרון גדול הוא להתמחות בתת תחום צר יותר כמו בדיקות אפליקציות, בדיקות תשתיות, אבטחת מידע, בדיקות בתחום הפיננסי וכדומה. התמחות כזו תעזור לכם לצבור ידע עמוק ומוניטין כמומחים.

4. פתחו כישורי תקשורת בין-אישית - בודקי תוכנה עובדים בצוותים רב-תחומיים ונדרשים לשתף פעולה עם מפתחים, מנהלי מוצר, אנשי תשתיות ועוד. פתחו יכולת להסביר ולנמק החלטות באופן ברור, להציג רעיונות ביעילות ולנהל דיונים מבוססי עובדות.

5. היו יצירתיים ונכונים לקחת סיכונים - לעתים כדי לפרוץ דרך חדשה ולייעל תהליכים, יהיה עליכם לחשוב מחוץ לקופסא ולהציע רעיונות חדשניים. אל תהססו לקחת יוזמה, לנסות גישות חדשות ולקחת סיכונים מחושבים.

6. גלו סקרנות, יושרה והקפדה - היו סקרנים לגלות ולחקור דברים חדשים, שאלו שאלות ותהיו אנליטיים. במקביל, שמרו על יושרה מקצועית ודייקנות בכל משימה שתבצעו, תנו דיווחים מדויקים והיצמדו לסטנדרטים ונהלי עבודה נאותים.

7. טפחו חשיבה חיובית - לעתים קרובות תיתקלו באתגרים, כשלים ובעיות מורכבות שיהיה קשה למצוא להן פתרון מידי. שמרו על גישה חיובית, סבלנות והתמדה, ותשאפו להמשיך וללמוד מכל מקרה. זו דרך מצוינת להתקדם מקצועית.

בתור בודק תוכנה מתחיל, השילוב של למידה מתמדת, התמקצעות,

ראיון עם מנהל בדיקות | אור אמויאל CYREBRO

גמישות ויצירתיות יוביל אתכם לקריירה מוצלחת ומלאת סיפוק בתחום חשוב ומאתגר זה.

ממה היית ממליץ להמנע?

כבעל ניסיון של 10 שנים בתחום בדיקות התוכנה, הייתי ממליץ לבודקי תוכנה מתחילים להימנע ממספר דברים:

1. מלהסתפק ברמת ידע בסיסית - תחום הבדיקות משתנה במהירות וחשוב להמשיך וללמוד באופן עצמאי כלים חדשים, טכנולוגיות ושיטות עבודה. אי למידה מתמדת תשאיר אתכם מאחור.
2. מחשיבה צרה/חד-ממדית - בתור בודקים עליכם לפתח חשיבה ביקורתית ואנליטית, ליצור מקרי בדיקה ותרחישים שחושבים מחוץ לקופסא ובודקים את המערכת מזוויות שונות.
3. מפחד לשאול שאלות ולחלוק ספקות - אל תהססו לשאול שאלות גם אם נראה שהן נאיביות. חשוב לחלוק ספקות עם עמיתים והצוות ולוודא הבנה מלאה.
4. מלקבל דברים כפשוטם ללא חקירה - אל תיקחו דברים כמובנים מאליהם. חשוב לבדוק ולאמת הנחות, לנתח דוחות ולהציף חריגים וממצאים חשודים.
5. מהתעלמות מכללים, נהלים ותקנים - עליכם לשמור על סטנדרטים, תהליכי עבודה נאותים ודרישות ציות רגולטוריות רלוונטיות בתחום שלכם.
6. מחוסר ארגון ואי-דיוק - בתור בודקים, אתם נדרשים לנהל בצורה מסודרת תיקי בדיקות, תיעוד הממצאים והעבודה וכן לשמור על דיוק בכל פעילות שאתם מבצעים.
7. להימנע מגישה שלילית ודעות קדומות - גלו גמישות, סקרנות ונכונות לקבל דברים חדשים. אל תסגרו דלתות בגלל חוסר ידע או אי נוחות. היו פתוחים לרעיונות חדשניים.

פספורט קבוצתי CYREBRO

סוג המוצר הנבדק: פלטפורמות ניהול ואינטגרציה של אבטחת מידע וניהול סיכונים סייבר לארגונים:

מוצר הדגל שלנו מאפשר לאסוף ולאחד באופן אוטומטי מידע ממקורות אבטחה מגוונים – חומות אש, מערכות SIEM, סורקים, משגרי התראות וכלים רבים אחרים. לאחר מכן, המערכת מנתחת את המידע הזה בהקשר המלא, על בסיס מאגרי הידע ומנגנוני ה-ML/AI שלה.

גודל קבוצת הבדיקות: 3 מפתחי אוטומציה ו-3 בודקים ידניים.

וותק הבודקים: וותק בודקי האוטומציה: בין 6-8 שנים | וותק הבודקים הידניים: בין 4-6 שנים.

סוגי בדיקות: כצוות בדיקות למוצר מורכב של ניהול יכוני אבטחה וסייבר, אנו נדרשים לעשות שימוש במגוון רחב של שיטות וסוגי בדיקות כגול: בדיקות פונקציונאליות, אינטגרציה, עומסים, אבטחה, כשלים ושחזור, בדיקות מקרי קיצון, ניידות ופלטפורמות, בדיקות גרסה ורציפות תוכנה

המראיין: ניצן גולדנברג
מזה 8 שנים בתחום בדיקות התוכנה, בתפקיד נוכחי מהנדס בדיקות בכיר בחברת Seatgeek.
מנהל קבוצת ה-AB של ארגון ITCB®
המוביל הראשי של קבוצת המיטאפ TestIL ומרצה בקורסים לבודקי תוכנה

מזל טוב לאלופי הבדיקות של ישראל

זו השנה השמינית לתחרות הבדיקות הישראלית ISTC שהתקיימה בזכות תמיכתם של - ITCB® העמותה הישראלית להסמכת בודקים, Matrix Testing & Automation, ג'ון בריס הדרכה, Inflectra, חברת Cloudbeat, PractiTest, Provengo, Jam.dev, ופארק הייטק צפון.

התחרות מיועדת לבודקים מנוסים וגם לאלו החדשים בתחום, המשוגעים לדבר אשר אוהבים אתגרים ונהנים לבצע בדיקות. במסגרת התחרות התבקשו המתמודדים לבדוק מוצרים אמיתיים בתנאים קיצוניים ובזמן מוגבל.

המשתתפים נדרשו לזהות באגים, לדווח עליהם באופן ברור המאפשר תיקון, לערוך בדיקות לא פונקציונליות לפי הצורך ולסיום גם להגיש דוח סיכום מצב המוצר אשר יהווה ערך נוסף לחברות המוצר מבחינת שחרור הגרסה.

במוקדמות שנערכו השנה בשלושה מועדים שנים: 08.05.24 | 10.05.24 | 15.05.24 השתתפו 45 צוותים ודווחו 361 תקלות.

בגמר שנערך ב-23 ליוני 2024 השתתפו 5 צוותים ונמצאו 64 תקלות.

צוות השופטים במוקדמות ובגמר כלל מנהלים ויועצי בדיקות בכירים המייצגים את חברות ה-IT וההייטק המובילות בישראל: **ירון צוברי** – ITCB®, **ניצן גולדנברג** – העורך הראשי של מגזין "עולם הבדיקות", יו"ר ומנהל התחרות, **משה מאמיה** – סמנכ"ל אבטחת מידע וענן ב-Totango, **אסתר צבר** - מקימת AQA, **קובי יונסי** - מייסד Kobiyonasi.co.il, **טל פאר** – חבר בהנהלת ITCB®, יועץ ומדריך בדיקות, **גיל שקל חבר בקבוצת AB של** ITCB®, **דני אלמוג** – מרצה וחוקר במכללה האקדמית סמי שמעון **וד"ר הדס חסידים** – מרצה בכירה במכללה האקדמית סמי שמעון.

פרסים יקרי ערך הוענקו לזוכים בגמר

מקום– I
טיסה ,אירוח והשתתפות בכנס Agile Testing Days 2024 שיתקיים בפוטסדאם, גרמניה בנובמבר 2024 בחסות:
ITCB®, PractiTest & Cloudbeat, Jam.dev & Provengo

מקום– II
קורס מקצועי (עד 40 שעות) בחסות מכללת ג'ון בריס ואוזניות Sony בחסות ITCB®

מקום– III
קורס מקצועי (עד 40 שעות) בחסות מכללת ג'ון בריס

תואר Best Bug Award במוקדמות ובגמר
שוברים בסך 500 ₪ כל אחד בחסות Letstop, Ivy מוצרי דיגיטל, בשביל החיים ו-Matific

העולים לגמר
זכו בכרטיס חינם **לכנס Testing & Automation GeekWeek 2024**

הזוכים של ISTC 2024

מקום ראשון: צוות Shamir Optical QA

אנחנו אברהם ואוריה, עובדים יחד בשמיר אופטיקה
השתתפנו בתחרות בשביל האתגר והזדמנות להכיר וללמוד.
כל יום זו הזדמנות למשהו חדש, מלא במחקר, בדיקות וכיף עם המון שיתוף פעולה.
ה'אני מאמין' שלנו הוא: כל אתגר הוא רק באמפר

אוריה ולדמן

אברהם בוטבול

מקום שני: צוות KoMamanov

עד לפני מספר חודשים, שנינו עבדנו יחדיו בחברת סיטגיק, ובמשך תקופה משמעותית היינו חלק מאותו צוות אוטומציה. טרם הגיענו לתחום ה-QA, עברנו הסבה ממקצועות שונים לחלוטין: עמרי שימש כשופט כדורסל ומורה לחינוך גופני, בעוד שאלכס היה שוטר. מעבר לקשר החברי המצוין שלנו, אנו חולקים מוטיבציה ודחף להמשיך ללמוד ולהיחשף לנושאים וכלים חדשניים.
אנו מאמינים שלמידה והתפתחות מקצועית הן ערכים חשובים בחיים. למרות שבאנו ממקצועות שונים לגמרי בעבר, אנו מונעים על ידי סקרנות ורצון להמשיך וללמוד דברים חדשים. אנו מחויבים להתקדמות אישית ומקצועית, ופתוחים להזדמנויות שמאפשרות לנו לרכוש ידע וכלים חדשים. היכולת להסתגל ולהתפתח היא חלק משמעותי מהגישה שלנו לחיים ולקריירה.

אלכס קומנוב

עומרי ממן

מקום שלישי: צוות GreenTest

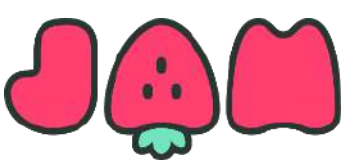
אני אוהב את תחום הבדיקות ונהנה מאתגרים מעניינים. התחרות פתחה לי דלת לאתגר מעניין והזדמנות להתפתח מקצועית ולימוד על מוצרים וטכנולוגיות חדשים

יהונתן אושרי דיוויס

ברכות גם לשאר הצוותים שעלו והתחרו בגמר גביע וזכו בתואר Best Bug Award

מורן להב ורינת בורנשטיין – צוות Truetesters – זכו גם בתואר Real Bug Award
אלכס אוסטרובסקי וניר ברונשטיין – צוות Smokers

	סטטיסטיקות המוקדמות	סטטיסטיקות הגמר
משתתפים	45 צוותים – 7 צוותי יחיד ו-38 צוותי זוגות	5 צוותים - צוות אחד של יחיד ו-4 צוותים של זוג
המוצר הנבדק	Letstop - בואו ננהג חכם Ivy – צאו מחוץ לקופסא בשביל החיים – חיפוש נפטרים	Matific – פלטפורמה חינוכית לכיתות א-ו
כמות תקלות שדווחו	361	64
כמות ממוצעת של תקלות שנמצאו לצוות	8	12
מקסימום של תקלות שנמצאו ע"י צוות אחד	42	19
מינימום של תקלות שנמצאו לצוות אחד	2	11



בחן את עצמך | ירון צוברי

זוכרים את התקלה הראשונה שדיווחתם עליה? אם כן, וואו, שאפו! אגב, האם הייתם מדווחים עליה היום באופן זהה או שונה?

אני, למען האמת, לא זוכר במדוייק את התקלה הראשונה שדיווחתי, אבל אני כן זוכר שהיא הייתה על מסך של VAX400, אם אני לא טועה... יותר מ-30 שנה... לך תזכור... זו הייתה טעות בחישוב שהובילה למסך אפור... בדיוק אחרי שהחלפנו את המסכים מירוק לאפור.

אני זוכר מספר דיווחי תקלות (שהפילו אותי לרצפה) ושיצא לי להיתקל בהן בראשית דרכי כמו: "הרולס-רויס שלי לא עובד!" – הכוונה הייתה לקורא השיקים שמריץ את השיקים ומבצע קריאה לפס המגנטי (או האופטי) שעל גבי המסמך; "המדחום שלי תקוע...!" – תוכנה להפקת דוח סטטיסטי השתמשה בתצוגה של התקדמות התקנה (מכירים? בכל פעם שאתם מתקינים תוכנה ישנו פס הבנוי ריבועים וככל שההתקנה מתקדמת, נוספים יותר ריבועים), רק במקום שתצוגת ההתקדמות תהיה אופקית, היא הייתה גם אופקית וגם אנכית (אכן נראה כמו מדחום דיגיטלי...). האם המפתחים אצלכם היו מקבלים היום דיווחי תקלות כאלו?

בודקים אחראים לאשר שהתוכנה שנבדקת אכן עובדת, אבל לא הכל עובד כצפוי. יותר נכון לומר, לרוב התוכנה לא עובדת כצפוי והבודקים - שלעיתים נתפסים כאחראים לאיכות התוכנה - נדרשים לדווח על הפגמים (defects, bugs) שמצאו בתוכנה באמצעות דוח פגמים (defect report). לכן, מאחר ואחת המטרות של בדיקות היא למצוא פגמים, יש לתעד פגמים שנמצאו במהלך הבדיקות. הדרך בה יתועדו הפגמים עשויה להשתנות בהתאם להקשר לרכיב או למערכת הנבדקת, לרמת הבדיקה ולמחזור חיי התוכנה. תיעוד ומעקב אחרי פגמים עשוי להיות מאוד לא פורמלי. בעיקר כיוון שניתן לדווח על פגמים בשלבים שונים של מהלך חיי הפיתוח, למשל, במהלך כתיבת הקוד, ניתוח סטטי, סקירות, במהלך בדיקות דינמיות או שימוש במוצר תוכנה. ניתן לדווח על פגמים בקוד או במערכות עובדות, או בכל סוג של מסמך, כולל דרישות, סיפורי משתמש וקריטריון קבלה, מסמכי פיתוח, מסמכי בדיקה, מדריכי משתמש או מדריכי התקנה.

השאלה של היום מציגה איך למקד את סיכום דיווח הפגם כך שיביא לידי ביטוי באופן מיטבי את מהות הכשל ואת מידת השפעתו על בעלי העניין.

(לשם הבהרה חשוב לציין שאני בוחר להשתמש במושג פגם, כמו שמופיע בסילבוס בעברית, למילה שבשפת היומיום אנחנו קוראים "באג" מאחר וה"באג" אותו אנו מוצאים בזמן הבדיקות הוא למעשה ה"כשל"). למידע נוסף, מומלץ לקרוא את פרק 5 תת-פרק 5.6.1 של תוכנית ההכשרה הבסיסית של ארגון ISTQB).

ועכשיו לשאלה (כתובה בלשון זכר, אך מתייחסת לכל המינים):

אתה עובד כבודק של מערכת בנקאית מקוונת. זמינות נחשבת לאחד מסיכוני המוצרים (איכות) העיקריים עבור המערכת. אתה מוצא כשל שניתן לשחזר, שגורם לכך שלקוחות מאבדים את התקשורת שלהם לאתר האינטרנט של הבנק בעת העברת כספים בין סוגי חשבונות נפוצים ואינם יכולים להתחבר מחדש במשך שלוש עד חמש דקות.

איזה מהבאים מהווה סיכום (summary) טוב לדוח פגמים (defect report) עבור כשל זה, כזה שתופס הן את מהות הכשל והן את השפעתו על בעלי העניין (stakeholders)?

א. יומני שרת אינטרנט מציגים שגיאה 0x44AB27 בעת הפעלת בדיקה 07.005, שאינה הודעת שגיאה צפויה במערכת הקבצים /tmp

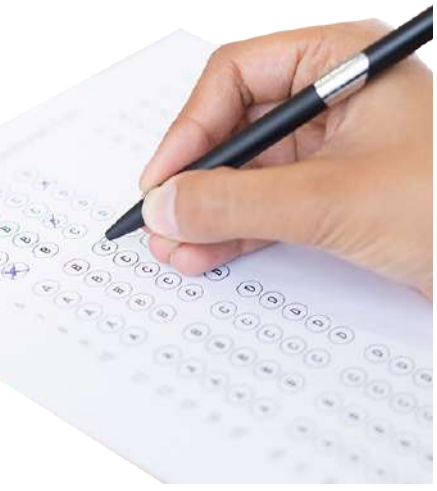
ב. מפתחים הציגו פגם גדול בזמינות שיעצבן את הלקוחות שלנו

ג. הביצועים איטיים והאמינות מתקלקלת תחת עומס

ד. עסקת העברת כספים אופיינית גורמת לניתוק הלקוח, יחד עם עיכוב בזמינות בעת ניסיון להתחבר מחדש

בחר אפשרות <u>אחת</u>

***לצפייה בתשובה המפורטת- דפדפו לעמוד 20**



ירון צוברי

מעל 30 שנות ניסיון בפיתוח תוכנה, ניהול פרויקטים מורכבים, הנדסת מערכות ובדיקות. עם ניסיון בינלאומי במערכות מורכבות בתחומים: טלקום, פיננסים, אוטומוטיב ומערכות הגנה. מומחיות עיקרית: ניהול פרויקטים מורכבים, ייעוץ להנהלה בכירה בארגונים (Siemens Germany), אימון מנהלים. נשיא וסגן.

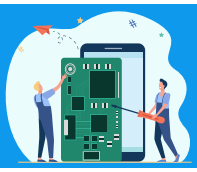
הצטרפו לצוות המוביל את מגזין עולם הבדיקות

מעוניינים להצטרף לעשייה? התפנה מקום בצוות המגזין!

הפעילים במגזין הינם אנשי מקצוע בתחום הבדיקות שפועלים בהתנדבות למען קהילת הבודקים בארץ.

לקבלת פרטים נוספים פנו לניצן גולדנברג:
magazine@testingworld.co.il

החשיבות בבדיקות זיכרון במכשירי אנדרואיד



טופז אנגלנדר

על אף שהן לפעמים סמויות מן העין, במאמר זה נצלול לתוך עולם בדיקות הזיכרון במכשירי אנדרואיד ונעמוד על חשיבותן

להשרות אמינות ומצוינות בתוכנה

בדיקות פונקציונליות מתמקדות בפעולות ובתכונות התוכנה, לעומת זאת הבדיקות הלא-פונקציונליות מתמקדות במאפיינים הלא-פונקציונליים של התוכנה, למשל: ביצועים, נוחות שימוש, יציבות, ואמינות.

התמקדות בבעיות ביצועים

בדיקות זיכרון משחקות תפקיד מכריע בסט הבדיקות הלא-פונקציונליות.

אחת הבדיקות החשובות במסגרת סט הבדיקות הלא-פונקציונליות היא בדיקת ניהול הזיכרון.
הסיבה לכך היא שבדיקות זיכרון הן כמו בדיקות תפקוד קוגניטיבי תקין של המוח: בדיקות שבהן מוודאים את היכולת לאחסן ולשלוף מידע בצורה אפקטיבית, באופן דומה ליכולת של המוח לנהל זיכרונות.

בדיקות הזיכרון מאפשרות לנו את היכולת לבחון את יכולת התוכנה לתת הקצאה ולשחרר הקצאה באופן יעיל למשאבי הזיכרון. כמו שאנו מארגנים את הזיכרונות שלנו בראש בצורה כזאת שתאפשר לנו לגשת בצורה מהירה ויעילה לזיכרונות. תחשבו על הבדיקות האלו כמו על בדיקות של מדען מוח שבודק את היכולת לשמור זיכרונות ולמחוק אותם, כל אלו מוודאים את יכולת התוכנה לתפקד בצורה חלקה ויעילה וללא עיכובים מיותרים או נפילות בביצועים. כמו שמערכת זיכרון תקינה חשובה וחיונית לבריאותנו, ככה ניהול זיכרון יעיל וחיוני בעולם התוכנה וקריטי לתפקודה התקין של התוכנה.

התעלמות מבדיקות זיכרון, עשויה לגרום לכך שתקלות חשובות מתגלות ב-Production ולא במהלך סבב הבדיקות. מדובר בתקלות קריטיות שיכולות להשפיע על ביצועי התוכנה, על חווית המשתמש ולגרום בסופו של דבר לקריסות.

> **"התעלמות מבדיקות זיכרון, עשויה לגרום לכך שתקלות חשובות מתגלות ב-Production ולא במהלך סבב הבדיקות"**

חשיבות האופטימיזציה ותשומת לב רבה לפרטים הקטנים: Garbage Collector מוגבל ביכולות לזהות את כל סוגי דליפות הזיכרון

נאמר שאנו בודקים אפליקציית אנדרואיד עם Activity שיוצרת באופן דינמי קבצי Bitmap ל-ImageView. כאשר ה-Activity נסגר (למשל: כי משתמש ניווט מחוץ ל-Activity הזה), ה-Bitmap Reference לא משוחרר. בתרחיש הזה, שה-Activity משוחרר, ה-Bitmap Reference מוחזק על ידי ה-ImageView והוא לא משוחרר.
ה-Garbage Collector עשוי לא לזהות את זה בתור דליפת זיכרון, היות שיש עדיין Reference שמחזיק את ה-Bitmap והוא באופן טכני ניתן להשגה. כתוצאה מזה, הזיכרון בשימוש על ידי ה-Bitmap עשוי לא להתפנות, זה יכול להוביל לדליפת זיכרון פוטנציאלית.

לעומת זאת, ה-Android Studio Memory Profiler יכול לזהות דליפת זיכרון מסוג זה היות שהוא יכול לזהות Instances שבהם ה-Object מוחזק שלא לצורך, אפילו כשיש להם Reference, ומזהה אותם בתור דליפת זיכרון פוטנציאלית.
אף על פי ש-Garbage Collector אמור לתת מענה לטיפול בזיכרון שלא בשימוש, ניהול זיכרון מדויק יותר נותר חיוני בשביל למנוע צווארי בקבוק. לכן, ביצוע של בדיקות זיכרון היא פרקטיקה מומלצת ואמצעי אקטיבי להבטחת יציבות המערכת וחווית משתמש טובה.

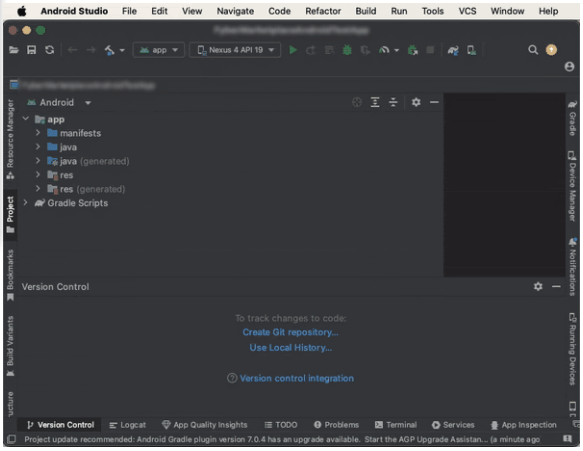
באמצעות ביצוע של בדיקות זיכרון, צוותי QA יכולים להבטיח כי התוכנה שאותה הם בודקים עומדת בסטנדרטים גבוהים ופועלת בצורה חלקה ואיתנה.

בעוד כלים אחרים כמו LeakCanary זמינים ויכולים בהחלט לסייע בזיהוי דליפות זיכרון, במאמר זה נתמקד ב-Profiler ובשימוש שלו ב-Android Studio.

Android Studio: כלי פיתוח שמאפשר גם בדיקות לניהול זיכרון המדגים את חשיבות בדיקות הזיכרון.

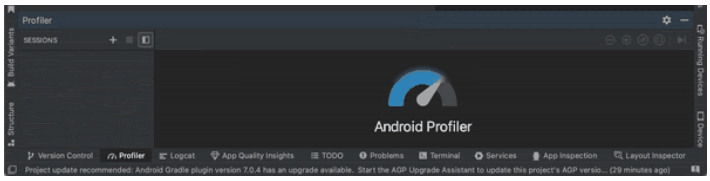
בואו נצלול לפרקטיקה באמצעות Android Studio וכלי ה-Profiler. באמצעות Android Studio נדגים את ההשפעה והחשיבות הרבה של בדיקות זיכרון במהלך הפיתוח של תוכנה.

פתחו ב-**Android Studio** את הפרויקט עליו את עובדים -> בחרו בתפריט **View** -> בחרו ב-**Tool Windows** -> בחרו ב-**Profiler**.

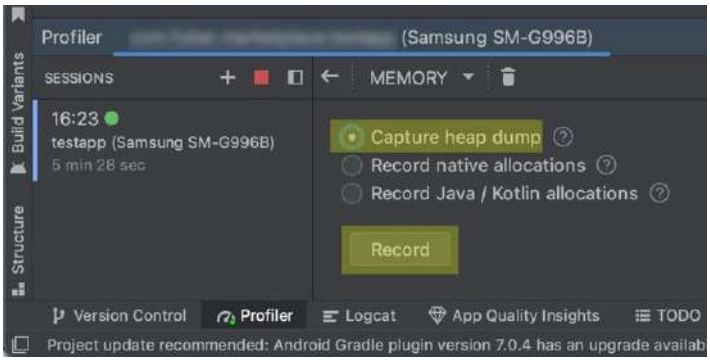


בחרו Profiler session חדש באמצעות לחיצה על כפתור ה-**+**, כעת בחרו במכשיר המחובר שלכם ואת ה-**package** הרלוונטי אותו אתם רוצים לבדוק.

נתמקד באופן ספציפי ב-Memory, בחרו ב-**Tab** של **Memory** שנמצא בתוך ה-Profiler.



לאחר שתפריט ה-Memory Profiler נטען, בצעו סט בדיקות והפעילו את אופציית ה-Memory Capture באמצעות לחיצה על כפתור **Capture heap dump** ואז על כפתור **Record**.

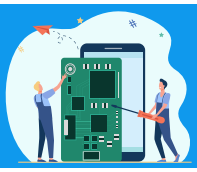


הפעולה הזאת שומרת תמונת מצב נוכחית של ניצול זיכרון האפליקציה, בעזרת מידע זה ניתן לאבחן את הזיכרון שבשימוש ברמת פירוט מדויקת. פעולה זה עוזרת במיוחד לזיהוי אובייקטים או משתנים ספציפיים שגורמים לבעיות בזיכרון.

טופז אנגלנדר
QA Engineer ב-Digital Turbine קרוב ל-5 שנים.

בוגר קורס בדיקות תוכנה וטכנאי מובייל. חובב מושבע של טכנולוגיה וסלולר בפרט, כותב ביקורות ומבקר מוצרים בערוץ יוטיוב וקבוצות פייסבוק.

הבעלים והמייסד של תוסף הכרום Tab Management

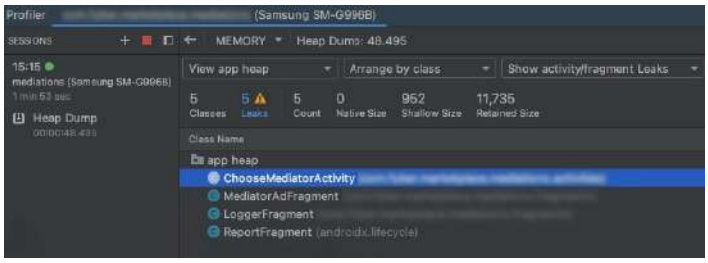


החשיבות בבדיקות זיכרון במכשירי אנדרואיד

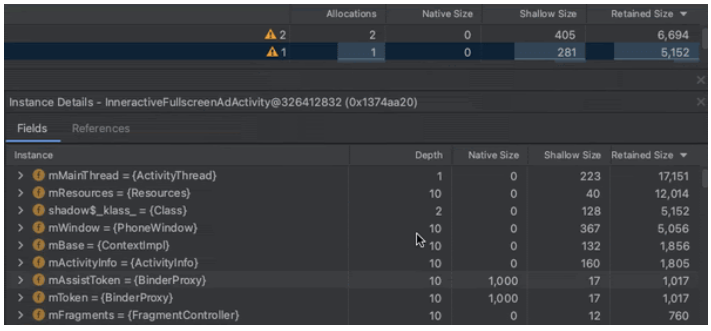
טופז אנגלנדר

בתרחיש שמודגם בצילום מסך הבא, הבדיקה עברה בהצלחה עם תוצאה של 0 דליפות מזוהות. במקרה הזה, הצלחנו לאשר את גמישות האפליקציה בניהול הזיכרון שלה.

בניגוד לכך, הצילום מסך הבא חושף תוצאה שונה לגמרי אשר עשויה להתגלות במהלך הבדיקות. כאן מדובר בכשל של הבדיקה כאשר מספר דליפות זיכרון המזוהות שהתגלו הוא 5. במקרה הזה קיבלנו תוצאה שונה מאת שציפינו לה, מה שמציף בעיה פוטנציאלית בניהול הזיכרון של האפליקציה.



השונות בין התוצאות מחייבת בחינה מוקפדת. מידע שימושי ביותר הוא ה-References וה-Fields אשר מושפעים מהדליפות המזוהות. המידע הזה שימושי מאוד למפתחים ומאפשר זיהוי ותיקון מהיר יותר של התקלה הפוטנציאלית. חשוב מאוד לתעד את המידע הזה בדיווח באג מקיף. תקלות של דליפת זיכרון בדרך כלל נחשבות לחמורות ובדרך כלל מטופלות במהלך הפיתוח טרם התוכנה יוצאת לשוק.

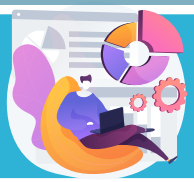


אילו סוגי בדיקות נבצע על מנת לאתר דליפת זיכרון?

שימוש בכלי לניטור זיכרון הוא טוב ויפה אבל בלי ביצוע תרחישי בדיקה מתאימים לא תהיה לו שום תועלת. חשוב להבין שאין כאן עשה ואל תעשה. אמליץ להתמקד בראש ובראשונה בתרחישי בדיקות נפוצים שאותן התוכנה אמורה לבצע (Sanity) זאת על מנת לשלול דליפת זיכרון בתרחישים בסיסיים. על מנת לנטר דליפות עמוקות יותר: אפשר לצלול לבדיקות עם יותר עומס על התוכנה, בדיקות של תרחישים שליליים יותר או בתפריטים יותר עמוקים בתוך התוכנה וגם בדיקות נסיגה (בהנחה והן רלוונטיות בשלב הזה של הפיתוח) עשויות להשלים מכלול מלא יותר לתוכנה בשלב הפיתוח. מניסיוני ניתן לנטר דליפות זיכרון במגוון רחב של תרחישים גם בסיסיים וגם עמוקים יותר.

שיפור אמינות אפליקציות באמצעות בדיקת זיכרון ב-Android Studio

באמצעות שילוב בדיקת זיכרון בשלב הפיתוח עם Android Studio, צוותי ה-QA יכולים למצוא מראש אתגרים ותקלות אפשריות, אנשי QA יכולים לבדוק הרבה יותר Test Cases שונים ומגוונים שאנשי הפיתוח לא בהכרח פנויים לבדוק אותם (וגם לא בהכרח מתפקידם לבדוק אותם) ובכך לוודא שהאפליקציה מפותחת באיכות גבוהה, עובדת בצורה יעילה ומשרה אמינות. השימוש בבדיקות אלו התברר בתור כלי אפקטיבי, המאפשר לנו לזהות ולטפל בבעיות באופן יעיל. כיום בדיקות אלו באמצעות Android Studio הפכו לחלק חיוני מסט הבדיקות שאנו עורכים באופן קבוע. אני ממליץ לכם להשתמש בכלי הזה, כמו גם למצוא את הכלי הרלוונטי לסביבת הפיתוח שבה אתם עובדים בשביל לערוך בדיקות מתאימות.



משולחנו של שביט
מה עלול לקרות כשבודק אחד מחזיק בכל הידע?

שביט ג'רסי

אבא לתאומים בני 3, מנהל בדיקות בחברת Wisetamp, בעל תואר ראשון בהנדסת תעשייה וניהול מהטכניון, בעל 11 שנות נסיון בתחום הבדיקות, מתוכם מעל 8 שנים מדריך עצמאי ל-QA.

בעל סדרת הסרטונים השבועית "QA ללא הפסקה"

חברים, שיהיה לכם רבעון מוצלח ושבוע טוב לכולם. אנחנו בשלהי הרבעון השלישי היום
ואת הכתבה הזו אני רוצה להקדיש לעובדים החדשים שהתחילו את עבודתם בצוותים שונים.

הפעם אני רוצה לדבר על מה שקורה במצבים שונים בהם אנחנו מגיעים למקום עבודה חדש כבודקים וגם בשלל תפקידים אחרים ועם הזמן מגלים שעובד איתנו מישהו שהוא אוטוריטה מסוימת לתפקיד מאוד נישתי והוא גם היחיד שמכיר דברים מסוימים במערכת ומתפעל כמעט לבד בעצמו מערכות מסוימות.
ממש כמו ה.."forward לשרגא" שאנחנו מכירים מ - "רמזור".

1. אבל אז אנחנו כפי שאתם מבינים בעצמכם, צפויים לבעיות שונות ומשונות..

   אותו בודק יחזיק בכל הידע בעצמו וידע לתפעל מערכות שונות בכמה דרכים ובאופן בלעדי.
   מצב זה לא בריא בכלל באף ארגון אבל הרבה פעמים קורה מתחת לאף של מנהלים שונים.

2. כשתהיה בעיה בין אם זה יקרה באוטומציה שנכשלת או נשברת וצריך לתפעל מערכות שונות, או בין אם צריך לתפעל מערכות אחרות כדי להזיז חלקים אחרים קדימה – אף אחד אחר לא ידע לעשות את זה אם למשל איש הצוות בחופשה, במחלה או יותר גרוע.. אוטוטו עוזב אותנו.

3. יותר גרוע מהכל – אותו הבודק מבין שהתפקיד שלו ייחודי למעשה ואז הוא ינסה בחלק מהמקרים להימנע מלחשוף בודקים אחרים לאותם מערכות ובאותה רמת ידע שהוא מכיר כדי לשמור את הידע והיכולות לעצמו מתוך אולי איזשהו חשש שבאיזשהו שלב לא יכירו בו כסמכות מקצועית לאותה פונקציה ספציפית.

ואז למעשה אנחנו, כבודקים בצוות או יותר מזה – כמנהלים - צריכים לשים לב לדברים האלו. צריכים לשים לב שהידע והיכולות מאוזנים ועוברים בין כל חברי הצוות ולא רק X יודע לעשות את זה. ו-Y יודע את זה. אלא כולם עושים הכל. כולם יודעים הכל.

כמובן שצריך לזכור שבמקרים מסוימים כשיש כמה וכמה אנשי צוות שבהם כל אחד מגיע עם מומחיות מסוימת ואז הדינמיקה תעבוד בצורה כזאת שכל בודק מתעסק זמן רב יותר במשימות או בכלים שהוא חזק יותר בהם ועם זאת אסור שנהיה תלויים, כלומר מצב שבו אנחנו לגמרי Clue-less ולא מבינים דבר ובודקים אחרים הם הבלעדיים להפעיל/להטריג ולנהל.

אחת מהיוזמות היותר טובות שאפשר לעשות במקרה כזה הוא להביא לידיעת מנהל הצוות את הדברים אם הוא לא ראה זאת בעצמו ואז יהיה ניתן לקיים Knowledge share שזה למעשה. שיחת שיתוף ידע בלייב עם כל חברי הצוות וכך אותה אוטוריטה לא יוכל יותר לשמור את הידע של אותו מוצר או נושא לעצמו בין אם הוא עושה זאת במקרה או בכוונה.

פגישה שכזאת תאלץ אותו לשתף בעל כורחו את הידע שהוא מחזיק עם שאר חברי הצוות ועצם השיתוף והפגישה יניבו הרבה שאלות בחזרה בכל נושא.

חשוב להבין כי במידע ופגישה זו לא תתקיים בזמן ויהיה צורך להמתין ל-Knowledge transfer כשאותה אוטוריטה מחליט לעזוב את הארגון ואז זה יקרה בד"כ בדקה ה-90, יתעוררו בעיות יותר קשות לצוות שנשאר.

זכרו – "שחקנים בודדים יכולים לזכות בנקודות – אבל רק קבוצה שלמה זוכה במשחק."

אני מאחל לכם המשך שבוע נעים שקט ורגוע
ושלעולם לא תצעדו לבד

עושים QA לקריירה
איך נראית השפעה שלא קשורה לטייטל שלך??

הצורך להשפיע על החלטות, אנשים ותהליכים משותף לכולנו, בכל תפקיד ובכל ארגון, מסכימים?

כשאנחנו בישיבות אנו מנסים להשפיע על התוכן, הדיון והאקשן אייטמס, כשאנחנו עושים דיזיין או קוד ריוויו - אנחנו רוצים להשפיע על האפיון ואיך שהקוד כתוב. אנחנו רוצים להשפיע על המנהלים והמנהלות שלנו - על ההחלטות, על איזה באגים ייפתרו, על מתודולוגיות העבודה ולא פחות חשוב על עצמינו - על איך שאנחנו תופסים את עצמינו ועל איך שאחרים תופסים אותנו.

אז איך משפיעים?

1. אפשר להשתמש בסמכות - מכח הטייטל שיש לכם, אתם מחליטים, אוכפים את ההחלטות ובכך משפיעים על אחרים
2. יש כאלו שמשפיעים מתוך המומחיות שלהם - כי הם מכירים הכל, כי הם היו שם מההתחלה וכל התושב"ע של המוצר אצלם בראש, ולא תמצאו אותו בקונפלואנס ("הדינוזאורים")
3. ויש את אלו שמכירים את כולם, את הארגון, איך דברים עובדים ויש להם מערכות יחסים מעולות שבעזרתן הם גורמים למה שחשוב להם לקרות

ואת כל אלו אתם מכירים, יש לכם קולגות כאלו, ואולי אתם מזהים באלו את עצמיכם. ואכן, במשך עשרות שנים, אלו היו מקורות ההשפעה הכי שכיחים בארגונים, לפי הסדר הזה.

אבל עולם עבודה משתנה, ואנחנו חלק ממנו, ויש בו Task Forces, צוותים רוחביים, תפקידים מטריציונים, וכולנו יודעים שבהייטק שינוי זה הדבר הכי קבוע,אז גם האסטרטגיה משתנה בקצב, לוקחת את הארגון קדימה ואנחנו - עושים אדפטציה ומיישרים קו.

בעולם העבודה הזה יש מקור ההשפעה אחר שצובר תאוצה, גם על חשבון הסמכות הפורמלית - וזאת השפעה מתוך מנהיגות!

בכל השנים שהובלתי קבוצות, או התקדמתי בארגונים, ידעתי שאני מנהלת טובה, הפידבקים היו מעולים, אבל מעולם לא העזתי לשאול - האם אני **גם** מנהיגה?

כי לימדו אותי, ואולי גם אתכם, שלהיות מנהיגה, אומר שיש לכם אוסף תכונות מיוחדות, כריזמה מתפוצצת, ושזה בכלל מולד בבחינת "או שיש לך את זה , או שאין לך את זה", אך היום אנחנו יודעים אחרת ואנו לומדים את זה גם ממה שאנחנו רואים בשטח מסביבינו בארגונים, וגם מהמחקר.

המחקרים העדכניים ביותר מצאו שמנהיגות היא רק 30% מולדת ו-70% נרכשת על ידי אימון, למידה והתנסות, כמו כל דבר שאתם מפתחים ומשכללים עד שאתם טובים בו.

"בעולם העבודה הזה יש מקור ההשפעה אחר שצובר תאוצה - השפעה מתוך מנהיגות!"

ומה שעוד אפשר למצוא במחקרים ובסקרים ארגוניים זה שמנהיגות פנימית, עצמית, היא תנאי כדי להנהיג אחרים ולרכוש אמון.

וזה די הגיוני - תחשבו, האם הייתם הולכים אחרי מנהיג או מנהיגה שלא יודעים להנהיג את עצמם? כנראה שלא.

אני מגדירה מנהיגות עצמית כך: "מלאכת השפעה והשראה שנובעת מבפנים." ככזאת, היא נובעת מתוך הכרה שיש לנו את הבחירה כיצד להתנהל אל מול נסיבות החיים שלנו.

מה שזה אומר, פרקטית:

- זה שאתם מבינים מי אתם - מה חשוב לכם? במה אתם חזקים? על מה אתם לוקחים אחריות מעשית, מה עושה לכם טוב ודוחף אתכם למעלה וגם, מה מוריד אתכם למטה וכיוון שמנהיגות היא מלאכת השפעה - אנחנו זה כלי העבודה.

המיקוד של הטור הזה, הוא בדיוק זה - ותוכלו להשתמש בכל כלי מהגיליונות הקודמים כדי להגביר מודעות לכל אלו.

- מנהיגות עצמית זה גם אומר שאתם מבינים לאן אתם מכוונים, ולמה אתם עושים את מה שאתם עושים?

ומהרגע שהבנתם את עצמכם, ואת הכיוון, אתם פרואקטיביים בדרך אל המטרות שלכם תוך כדי שאתם **מנהלים את העולם הפנימי שלכם בהצלחה.**

זה שריר שלומדים לפתח עם הניסיון, ההתנסות ובעיקר המוכנות לקחת אחריות ובעלות על הדרך שלכם בקריירה ובכל תפקיד בחיים.

בעברית להנהיג זה מהמילה נהג.

הנהג הוא זה שאוחז בהגה, הוא מכוון וקובע את הדרך. אין מישהו אחר שינהג עבורו את הדרך. בהקשר של פיתוח הקריירה זה אומר שאתם לא מחכים למנהל או המנהלת שלכם, כדי שאלו יפתחו את הקריירה שלכם, או להצעה שתנחת לכם במייל רק כדי לנקוט יוזמה. אלא, שאחרי שביררתם עם עצמכם מהם המטרות שלכם, ואתם יודעים לאן אתם מכוונים, אתם המנהיגים.

אז מאיפה מתחילים?

מהמקום שמי שאוחז בהגה מתחיל - לדעת מה היעד.

מנהיגות עצמית מתחילה מחזון, מטרות ופעולות. בכל הקשור לפיתוח הקריירה והמנהיגות העצמית, אנחנו מייצרים לעצמינו תמונת עתיד ובה מתארים מה חשוב לנו שיתממש בה. מה דוחף אותנו אליה, למה ובעבור מה (What for? Why?)

חזון - זאת אמנם מילה גדולה, ואולי אתם חושבים שזה משהו שרק מנהלים בכירים צריכים לעשות, אז לא. חזון לא קשור לתפקיד שלכם, הוא קשור לתפקוד שלכם.

אז לטובת מי שכבר התנסה בלייצר לעצמו כזה וגם לטובת מי שלא.

הנה תרגיל שמתאים לכולם:

הערב או מחר בבוקר, קחו לכם 20 דק׳. בדקות האלו אתם תהיו בשקט עם עצמכם ויהיה לכם גם משהו לכתוב בו - מחברת, לפטופ.

דמיינו לרגע שעברה שנה מהיום בו אתם קוראים את הטור הזה, אתם עכשיו מתישהו ב-Q3 2025, מרגישים את הקיץ, את אווירת ההתחדשות ואתם לוקחים זמן ומסכמים לעצמכם את השנה שהייתה לכם.

איילת מלמד כהן

בעלת ניסיון של 19 שנה בעולם ה-QA, רוב השנים כמנהלת בכירה בסטארטאפים וחברות גדולות. בשנים האחרונות מאמנת לפיתוח מנהיגות עצמית וניהולית, שיפור מיומנויות ניהול, כניסה לתפקיד חדש או ההתמודדות עם אתגרי התפקיד הקיים. בנוסף לאימון, איילת מלווה סטארטאפים בכל הקשור לניהול איכות, בונה צוותים ואסטרטגיות QA מותאמות לארגון. מאמנת מוסמכת בינ"ל, בעלת תואר ראשון ושני במנהל עסקים וכלכלה מאוניברסיטת בר-אילן. אמא לשלושה וזוקפת לזכותם חלק גדול ממיומנויות הניהול שלה.

עושים QA לקריירה
איך נראית השפעה שלא קשורה לטייטל שלך??

מה קרה בה, מה השתנה בכם? מה התפתח? איפה גיליתם אומץ?

איזה הישגים קרו? מה קרה לטובה והפתיע אתכם?

"חזון לא קשור לתפקיד שלכם, הוא קשור לתפקוד שלכם

כמה כללים לתרגיל:

1. לכתוב כאילו זה קרה בלשון עבר ולזרום עם הכתיבה, בלי לעצור, בלי לתקן, פשוט לכתוב. לדוגמא השנה הפכתי לארכיטקטית הבדיקות בארגון והובלתי מתודולוגיה חדשה לכתיבת טסטים. הפתיע אותי כמה מהר חברי הצוות אימצו אותה והפידבקים החיוביים שקיבלתי עליה, גורמים לי סיפוק גדול.
2. לשלב כמה תחומים מהחיים, לא רק מה קרה השנה בקריירה - תכתבו גם כמו מה קרה בזוגיות, ביחסים שלכם עם הילדים, בתחביב שלכם, או פיזית, לדוגמא - השנה עברתי לבית חדש ויש לו בו חדר עבודה מרווח עם הרבה עציצים, וליוויתי את הבן שלי בטיול השנתי.
3. לפרט - כמה שיותר פרטים על מה קרה לטובה, איך אתם מרגישים לגבי זה לדוגמא - אני מרגישה סיפוק, וגאה בעצמי על המאמץ, בזכות זה שהובלתי את מתודולוגיות כתיבת הטסטים, אני שמה לב שהרמה של הבדיקות השתפרה ומאז אני חונכת את כל העובדים החדשים עליה.

כל אחד כותב את מה שנכון ומתאים לו.

אם אתם מתקשים בכתיבה או שלא מתאים לכם, שבו עם מישהו שאתם סומכים עליו ותבקשו ממנו עזרה בלייצר לעצמכם מטרות.

תבקשו שישאל אתכם את השאלות האלו והנה עוד כמה שאני משתמשת בהם באימון מנהלים:

- איזו השפעה היית רוצה שתהיה לך על אחרים?
- באיזה חלק בתפקיד שלך, היית רוצה להיות מומחה?
- מה בתפקיד שלך הייתי רוצה שיהיה יותר או פחות?
- שנה מהיום, מה יראו אותך עושה אחרת?

חזון ומטרות - עוזרים להשפיע על הקריירה שלכם בכך שהם עוזרים לכם להכיר את עצמיכם, מה חשוב לכם, ולכוון את המאמצים שלכם. כשאתם מנהלים, החזון שלכם כולל את הצוות שלכם והמשימה. כשהוא ברור לכם, אתם יכולים לדבר אותו ואפשר להתחבר אליו ואליכם.

אחרי שאתם כותבים את החזון, כתבו לעצמכם 3-5 דברים שאם קרו בשנה הזאת, ללא קשר לתוצאה שאתם שואפים להשיג (כמו כסף, תפקיד, אחריות וכו), זאת עדיין תחשב מבחינתכם הצלחה. ותהיו כמה שיותר ספציפיים בהגדרה - אלו יהפכו להיות המטרות שלכם.

לדוגמא - אם אני QA בצוות אחד ובחזון שלי אני ארכיטקטית בדיקות של הארגון כולו. אז הצלחה מבחינתי תהיה אם אני אכיר את כל ה-Flows והארכיטקטורה של כל המוצרים של החברה.

או, אם בחזון שלי, אני רואה את עצמי מנהלת QA בארגון שונה מזה שאני בו היום, אז להיות מחוברת ומקושרת בארגונים שמעניינים אותי בתעשייה, ייחשב מבחינתי הצלחה.

3-5 לא צריך יותר. עם הזמן, אתם תעדכנו אותם ותשדרגו. אבל זאת נקודת ההתחלה האפקטיבית ביותר.

מחקרים הוכיחו שכשהחזון והמטרות ברורים לכם וכתובים, הסיכוי שתגיעו אליהם קופץ. אגב, זאת גם הסיבה שבכל ארגון אתם רואים תהליך של הגדרת יעדים, מטרות, KPIs OKRs רבעוני, חצי שנתי ושנתי.

השלב האחרון הוא השלב שמחבר את המטרות לעולם האמיתי וזה = תוכנית פעולה.

אתם לוקחים מטרה מטרה ושואלים - מה הפעולה האחת שאתם יכולים **ורוצים** לעשות כדי שהמטרה שלכם תתקדם?

והדגש הוא על מה אתם **באמת** רוצים לתרגם לפעולה, כי זה המקום שבו ההרגלים שלכם והמוטיבציה יסתכלו עליכם חזרה בראי ויאתגרו.

נחזור לדוגמא עם ה-QA שהחזון הוא להיות ארכיטקטית, אז הפעולה שאפשר לעשות זה להכין רשימה של כל הפיצ'רים שלא חזקים בהם, וללמוד אותם לעומק, אחד אחד.

כדי לעבור לניהול QA בארגון אחר, הפעולה תהיה: ליזום מפגשי קפה עם אנשים שעובדים בארגונים שמעניינים אתכם.

ומהרגע שיש את אלו, יש מיקוד. אבל גם התנגדות לשינוי, מי שקורא את הטור הזה, כבר יודע שהמוח שלנו עושה כל מיני טריקים כדי שנשמר את הקיים.

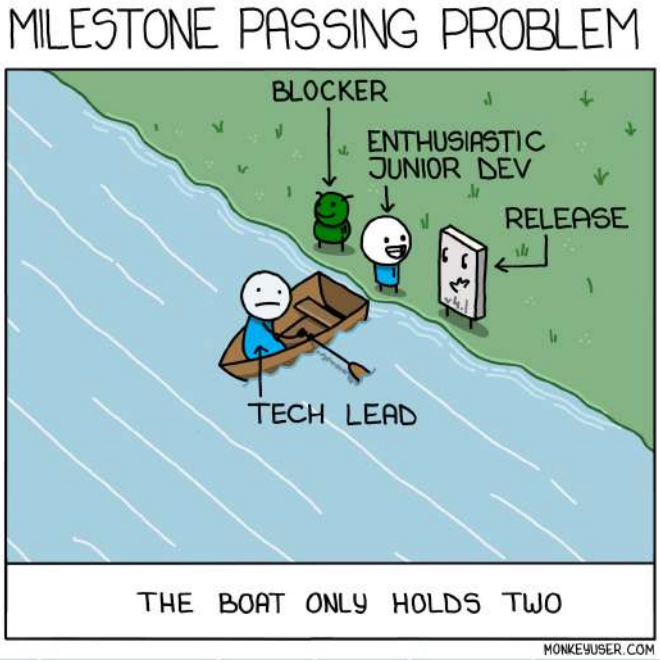
אז כדי להתגבר וכן לפעול למען המטרות שלנו, אנחנו צריכים להשתמש במודעות ולייצר לעצמינו סוג של פעולת דחף בעד עצמינו.

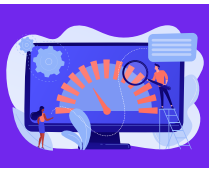
אני מתמודדת עם זה בעזרת פגישה מחזורית שיש לי אחת לחודש, ובה אני מזכירה לעצמי מחדש למה החזון שלי הוא דווקא זה, אני פותחת את הקובץ עם החזון והמטרות שיצרתי בתרגיל שמתואר למעלה, ואז אני מעדכנת את המטרות. אני שואלת את עצמי על כל מטרה מחדש - מה הפעולה האחת שאני **רוצה** לעשות כדי שהמטרה הזאת תתקדם? ואז מבצעת, או נועצת אותה ביומן.

מצאתי שהשיטה הזאת יעילה עבורי, אך הניסיון מלמד שיש כאלו שיותר מתאים להם לכתוב מחדש בכל חודש את החזון והמטרות, ואז האנרגיה שנוצרת מתוך התרגיל רותמת אותם לפעולה . יש כאלו שנעזרים במנטור.ית או בחבר טוב שישמש עבורם תזכורת אנושית למחויבות שלהם כלפי עצמם.

לכל אחד יש את הכלים שמתאימים לו, והתרגיל הוא פשוט לנסות.

האנשים שסובבים אותנו, כל הזמן צופים בנו, מתמודדים, שומעים מה אנו אומרים, רואים מה בפועל אנחנו עושים, אנשי הצוות שלכם רואים מה אתם עושים או לא עושים, על מה אתם לוקחים אחריות, ואיך אתם מתנהלים כשמשהו משתבש, אך גם אנחנו צופים בעצמינו. וכשאתם תצפו בעצמכם, יוצרים לעצמכם חזון, לוקחים אחריות על התפקיד שלכם ומתנהגים כמו האדם שאתם שואפים להיות - אתם תתפסו את עצמכם כמנהיגי חייכם וזה כלי ההשפעה החזק ביותר שיש לכל אחד כדי להשפיע בתוך הארגון ומחוץ לארגון, עם טייטל או בלעדיו.





בדיקת ביצועים ועומסים עם JMETER או K6 בעידן AI/ML | שי גינזבורג

במאמרים הקודמים שפרסמתי ברשת ובמגזין זה, הסברתי לקהל בודקי ומפתחי התוכנה את העקרונות של בדיקות העומסים והביצועים ואת המתודולוגיה שנוצרה על-מנת לתמוך בארכיטקטורה של שרתים ולקוחות הטרוגניים. המאמר הזה מוסיף לכל גוף הידע שהועבר כבר גם מידע חדש על הגרסאות המעודכנות של JMETER בהשוואה לכלי הצעיר K6. בנוסף, יש התייחסות לזרם החדש של AI/ML בקונטקסט של ביצועים ועומסים. אני מאחל לקהל קריאה מהנה ומעוררת מחשבה.

הקדמה

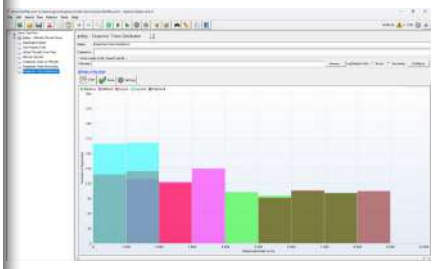
בהרצאות הקודמות שלי בנושא בדיקות עומסים ביצועים במסגרת המפגשים של קבוצת המיטאפ של TestIL (המפגשים התקיימו בבאר שבע, רעננה וחיפה) פגשתי קהל מגוון, שהביע עניין רב בנושא והשאלות שלא זכו למענה בגלל קוצר הזמן, קיבלו מענה אישי דרך המייל שלי sginsbourg@gmail.com. כמו כן, סיכמתי בכתב את הנושאים אשר הוצגו במפגשים, ואני שמח להגיש אותם כאן לקהל הקוראים של המגזין. אתם מוזמנים להמשיך להפנות אלי שאלות באמצעות הדוא"ל. תודתי נתונה לכל אלה אשר עמלו בהתנדבות מלאה על-מנת לקיים את המפגשים המוצלחים ברחבי הארץ וכמובן גם לכל אלה, שבזכותם יש לקהילת התוכנה מגזין נגיש ומעודכן בשפה העברית. יישר כוחכם.

רקע

כאשר משווים שני כלים חזקים מאוד לבדיקת ביצועים ועומסים Apache JMeter ו-K6, כדאי קודם לייצר מתודולוגיה לביצוע ההשוואה. קודם כל, יש להבין את החוזקות של כל אחד מהם, אחר כך נרצה לבדוק את ה-Use cases שלהם, כלומר לבדוק את ההתאמה של כל אחד מהכלים אל המקרים השונים בתעשייה והפרויקטים השונים, ולבסוף, נרצה לבדוק גם מהן הפשרות שאנחנו עושים, כאשר אנחנו בוחרים בכלי אחד במקום השני, כאשר אנחנו מתחילים לעבוד על פרויקט חדש. לפני שנתחיל את ההשוואה, ראוי שנציין, כי לשני הכלים יש פונקציות חזקות מאד והם מתאימים לפרויקטים רבים אך שונים. הבחירה בין JMeter ל-K6 תלויה בהקשר הספציפי, בדרישות ובמומחיות של הצוות אשר עתיד לבצע בפועל את הפרויקט. לשני הכלים יש כבר מקום חשוב בתעשייה, והבנת החוזקות שלהם היא הדרך הנכונה לקבל החלטות בנושא התאמת כל כלי אל פרויקט ספציפי. בואו נתעמק בנקודות המפתח, ונתחיל קודם עם Apache Jmeter.

הצגת הכלים המתמודדים והיכולות שלהם

JMeter הוא כלי (תוכנה בקוד פתוח) לבדיקת עומסים שבאמצעותו ניתן למדוד את ביצועי השרתים של אתרים ואפליקציות תוך כדי הפעלת עומסים ברמות שונות. העומסים נוצרים על ידי בוטים מתוכנתים אשר מחקים את הפעילות של גולשים אמיתיים באתר או באפליקציה שאנחנו בוחנים. Apache Jmeter כולל את כל ה-UI הנדרש על מנת להקליט את הבוטים, לעבד את ההקלטות, להריץ את הבוטים שהם הקלטות משודרגות ומשוכללות מאוד, וגם לנתח את התוצאות בזמן ובעיקר לאחר ביצוע ההרצה. האפליקציה JMeter כתובה בשפת התכנות Java, מה שהופך אותה לבלתי תלויה בפלטפורמה ונגישה באופן נרחב. במקרים רבים אפשר להקליט וגם לתכנת בנוחות רבה את הבוטים על לפטופ בסביבת Windows ואחר כך להריץ את אותם בוטים מתוך שרתים בסביבת Linux על גבי ענן פרטי, ציבורי, או היברידי. האפליקציה JMeter מספקת לנו את כל ה-UI שמתורגם לפקודות של Java, כלומר העומס נוצר מתוך פקודות של הקוד גם אם אנחנו לא יודעים לתכנת בשפה זאת, אבל יודעים להשתמש בכל ה-UI של Jmeter. הכלי הזה מתאים גם לבדיקות אוטומטיות פונקציונלית ובדיקות רגרסיה, שאינן קשורות לממשק הגרפי בצד הלקוח שהוא הדפדפן הרגיל שלנו ברוב המקרים, כי בבדיקות עומסים אין דפדפן שעושה Rendering בתוך חלון, וכל בוט צריך לחיות בתוך מרחב בזיכרון שאין לו פנים גרפיות. לא נוכל לקיים סוג של קשר עין עם כל בוט בגלל שאנחנו רוצים להריץ אלפים כאלה בו זמנית. כל בוט הוא Thread נוסף שמנוהל בתוך מכונה וירטואלית Java Virtual Machine אשר מתקיימת על גבי מערכת ההפעלה של הפלטפורמה המארחת. לשם השוואה והבהרה, כל Web Browser תופס כמעט Core שלם על מנת לפעול, וכמובן שאם אנחנו רוצים לבדוק למשל מיליון יוזרים הפועלים בו זמנית, לא נוכל להקצות משאבים הכוללים מיליון Cores לטובת אף בדיקה, כי זה אינו מציאותי, ולכן ניפרד כבר כאן כידידים מהדפדפן כאשר אנחנו נכנסים לעולם בדיקות העומסים.



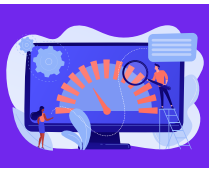
תמונה 1 – הממשק של JMeter בסיסי מאד אז זהו המעט שמחזיק את המרובה

המטרה המקורית בפיתוח הראשוני של JMeter הייתה הרצת בוטים בפרוטוקול HTTP בלבד לצורך בדיקת אתרי אינטרנט פשוטים, אך מאז הכלי התרחב מאוד מבחינת תמיכה בפרוטוקולים רבים אחרים (כולל JDBC, FTP, ועוד) ונוספו גם פונקציות משוכללות, דו"חות, תיעוד רב, ויכולות בדיקה רבות אחרות. JMeter מצטיין בהדמיית עומסים כבדים על יישומים כדי להעריך את הביצועים שלהם תחת עומסים כבדים. מעבר לבדיקת עומסים, הוא יכול גם לאמת את נכונות הפונקציונליות של האפליקציה, והוא מתאים מאוד לבדיקות רגרסיה פונקציונלית. הרצת הבדיקה יכולה להיות מבוזרת, כלומר אפשר לבצע בדיקת עומסים מתוך מספר שרתים בו-זמנית, מה שהופך את הכלי למתאים במיוחד לבדיקת תרחישים בקנה מידה גדול ממש. JMeter כולל ממשק משתמש גרפי (GUI) ליצירה, הקלטה, הרצה וניהול של תוכניות בדיקה. הממשק מיושן ובסיסי מאד, אבל הוא מקצועי ואמין, כך שבודקים יכולים לעצב תרחישי בדיקה באופן ויזואלי, להוסיף דגימות, להגדיר מאזינים Listeners ולהגדיר נקודות בדיקה Assertions, ועוד. הסקריפטים של JMeter הם קובצי XML, שניתן לערוך, לשלוט בגרסה שלהם, ולשתף בין צוותים. מבחינת תוספים והרחבה, JMeter בעל מערכת אקולוגית עשירה של תוספים והרחבות רבות, אשר זוכות לעדכונים מדי מספר חודשים. משתמשים יכולים לשדרג את הפונקציונליות על ידי התקנת תוספים נוספים או כתיבת תוספים חדשים לאחר התאמה אישית. מבחינת קהילה ותיעוד, ל-JMeter יש קהילה פעילה התורמת לפיתוח ולתחזוקה של הקוד הפתוח. קיים באינטרנט תיעוד רב ומקיף, הדרכות ופורומים זמינים למשתמשים. JMeter נמצא בשימוש נרחב לבדיקת עומס יישומי אינטרנט, ממשקי API ושירותי אינטרנט. הוא יכול גם להתחבר ישירות אל בסיסי נתונים אשר תומכים בפרוטוקול JDBC ואז לדמות שאילתות ישירות אל מסד נתונים, ולמדוד זמני תגובה בלי לעבור דרך כל הטופולוגיה ודרך שרתי האינטרנט. JMeter משתלב למעשה עם כל כלי ניטור לאיסוף מדדי ביצועים.

לסיכום, Apache JMeter הוא כלי רב עוצמה ורב-תכליתי לבדיקת עומסים ומדידות ביצועים, במיוחד במערכות אינטרנט ובמערכות מבוזרות. הכלי מבוסס על Java והוא צורך לא מעט זיכרון, במיוחד עבור בדיקות גדולות. הגמישות שלו, ההרחבות והקהילה הפעילה שלו תורמים לפופולריות שלו בקרב בודקים ומפתחים. בעוד שה-GUI

שי גינזבורג

מהנדס מכונות, תוכנה, וביו-רפואה בעל תואר שני בטכניון ותואר שני נוסף בהצטיינות באוניברסיטת תל-אביב. יועץ עצמאי לחברות היי-טק, סטארט-אפים, חברות ביו-רפואה, גופים כלכליים ומוסדות ממשלתיים בנושא בדיקות עומסים וביצועים של מערכות אינטרנט. בעברו היה מנהל QA של LOADRUNNER - מוצר הדגל של חברת מרקורי הישראלית אשר נכללה במדד נסד"ק 100. מרצה בפקולטה להנדסת תוכנה של המכללה האקדמית בבאר שבע ע"ש סמי שמעון. נולד ומתגורר כיום ביפו. מרבה לעסוק בתרגול יוגה, גידול כלבים, כתיבה, גידול צמחים למאכל ולנוי, ומוזיקה קלאסית ומודרנית.



בדיקת ביצועים ועומסים עם JMETER או K6 בעידן AI/ML | שי גינזבורג

מפשט את ההכרות הראשונית עם הכלי, הרי שפיתוח שליטה בתכונות מתקדמות דורשת זמן וכמובן שיש עקומת לימוד משמעותית.

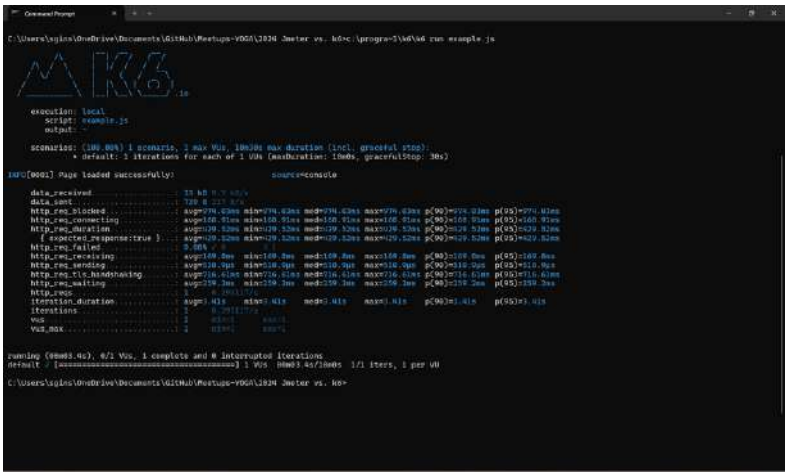
עכשיו נתעמק במתמודד השני שלנו...

K6 הוא כלי בדיקת עומסים ממוקד מפתחים. גם כלי זה נכתוב בקוד פתוח והוא מיועד לבדיקות ביצועים והערכת מדרגיות (Scalability), כלומר בדיקות ביצועים ברמות מדורגות של עומסים אשר הולכים וגדלים משלב לשלב. בניגוד לכלים אחרים, K6 כתוב בשפת התכנות Go (Golang), ואינו דורש סביבת Java או הקמת מכונת Java וירטואלית על מנת להריץ אותו - עובדה שתורמת ליעילות הכלי ולביצועים שלו, כלומר, מבחינה תיאורטית לפחות, אנחנו אמורים להפיק עומס רב יותר על גבי חומרה זהה בזכות העובדה, שהכלי המחולל את העומס מתוכנן באופן חסכוני למדי, לכאורה. K6 מעניק עדיפות לחוויית המפתח, מה שהופך אותו לכלי מועדף על משתמשים בעלי ידע טכנולוגי. כלומר, חוסר ה-GUI שלו מונע מאנשי QA רבים לנסות אותו בכלל, כי קשה לציבור הזה להתמודד עם כלי שאין לו חלון ידידותי, אלא דורש הקלדה של פקודות ארוכות בממשק ה-Terminal. מצד שני, המפתחים של הכלי הזה מכוונים אותו כנראה בעיקר לבדיקות אימות פיתוח, כך שהקמת GUI למוצר שלהם אינה נמצאת ככל הנראה ברשימת המשימות שלהם. ישנם היבטים מרכזיים של K6 שאנחנו חייבים להכיר, למשל, בניגוד ל-Jmeter, שהתסריטים שלו הם קבצים סטנדרטיים של XML, אז סקריפטים של K6 נכתבים בשפת JavaScript. מפתחי התסריטים יכולים ליצור תרחישי בדיקה אשר ניתנים לשימוש חוזר ומודולרי. בנושא ביצוע מקבילי (בו-זמנית) של משתמשים וירטואליים (Virtual Users), K6 מריץ מספר איטרציות במקביל, וכך הוא מדמה VUs. VUs פועלים כמו לולאות מקבילות בזמן, ומבצעות תרחישי בדיקה במקביל.

מבחינת התמקדות בתרחישים של גולשים אמיתיים - K6 בנוי לבצע תרחישים מציאותיים של בדיקת עומס. מפתחים יכולים לפרק בדיקות גדולות לחתיכות קטנות יותר לשימוש חוזר. ראוי לציין גם, שבניגוד לבדיקות המבוזרות עם JMeter - הבדיקות של K6 מבצעות מתוך מכונה אחת.

זה אמנם מפשט את ההגדרות והביצוע של ההרצות, אבל זה בהחלט אינו יתרון תחרותי מבחינת תקרת הביצועים כמובן. מבחינת מקרי שימוש, הרי שהכלי K6 מתאים היטב לבדיקת יישומי אינטרנט, ממשקי API ומיקרו-שירותים , וזה די חופף לכל מועמד אחר שנבדוק בעצם. מבחינת אינטגרציה מתמשכת (CI/CD), K6 משתלב בצורה חלקה עם תהליכים כאלה, וכך המפתחים יכולים להפוך את בדיקות הביצועים לאוטומטיות, כלומר לקבוע אותן כחלק מתהליך העבודה בפיתוח שלהם. הניסיון מראה, שהתכונה הזאת אינה רבת משמעות, מכיוון שההבדלים בביצועים בין גרסה לגרסה הם זניחים, אלא אם כן בוצע שינוי מהותי בארכיטקטורה של האפליקציה.

למעשה, נכון לבדוק ביצועים ועומסים מספר פעמים בפיתוח, אבל להטמיע את זה לתוך ה-CI זה לא תמיד רעיון מוצדק מבחינת ניצול משאבים הגיוני ברמת ניהול הסיכונים בפרויקט פיתוח. ל-K6 יש קהילה הולכת וגדלה של משתמשים ותורמים. קל למצוא באינטרנט תיעוד רב ומקיף ומדריכים זמינים למשתמשים. הקהילה כמובן צעירה וקטנה יותר בהשוואה לזאת של JMeter.



תמונה 2 – K6 חסר ממשק גרפי GUI אך עמוס ביכולות של בדיקת עומסים לתוכנה

לסיכום, החוזקות של K6 נעוצות בגישה הידידותית למפתחים, בביצוע מקבילי ובהתמקדות בתרחישים מציאותיים. זוהי בחירה מצוינת עבור יישומים ומיקרו-שירותים מודרניים. בניגוד ל-JMeter, ל-K6 חסר ממשק משתמש גרפי, וכמו-כן, המשתמשים חייבים להיות בעלי יכולת עבודה עם סקריפטים מבוססי קוד בשפת JavaScript. בעוד ש-K6 תומך בפרוטוקולים נפוצים (HTTP, WebSocket), הוא לא מכסה בינתיים את כל הרשימה הנרחבת של JMeter.

הכרות מעשית עם חוזקות הכלים לבדיקות עומסים וביצועים

כדי להפיק תועלת ממשית מהמאמר הזה, בואו נתקין את שני הכלים ונבנה ביחד תסריט ברמת בדיקת Sanity. המשימה שלנו תהיה לבדוק דף בית של אתר Demo. אפשר למשל, להשתמש באתר הבא לצורך אימון ולימוד:

https://demo.guru99.com/test/newtours/

כדי להתקין Jmeter, נכין קודם מכונת Java – קל מאוד להוריד גרסה מעודכנת מהאתר:

https://www.java.com/en/download/

ואז רק לפתוח קובץ ZIP מהאתר של Jmeter:

https://jmeter.apache.org/download_jmeter.cgi

לאחר ההורדה, פותחים את קובץ ה-ZIP, ואם אתם עובדים למשל בסביבת חלונות, תמצאו בתוך תיקיית BIN את קובץ ההפעלה jmeter.bat. כעת נתקין גם את K6 בקלות רבה. פותחים חלון Terminal ובתוכו כותבים את הפקודה

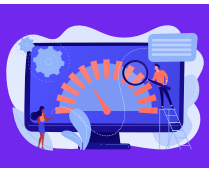
winget install K6 --source winget

כדי לתרגל בקלות את שני הכלים, הכנתי עבורכם דוגמאות שקל להוריד ולהריץ על המחשב שלכם לאחר ההתקנה. הקישור להורדות הוא:

https://github.com/sginsbourg/jmeter-vs-k6-2024

לאחר ההתקנה של הכלים וההורדה של הדוגמאות, נסו להריץ אותם ואף לשדרג את התסריטים. למשל, נסו לבדוק כיצד ניתן להוסיף Assertion שמוודא שאכן התקבל סטטוס OK 200 בכל אחת מהדוגמאות. אבל אל תעצרו כאן, כי יש שרתים ששולחים סטטוס הצלחה גם יחד עם עמודי שגיאה (מכיוון שעמוד השגיאה נמסר בעצמו מהשרת ללקוח בהצלחה). לכן, נסו להוסיף ולידציה נוספת אחרת וחד-משמעית, לדוגמא, מחרוזת טקסט ספציפית שלא תופיע בעמוד שגיאה בשום אופן. מצד שני, נסו אפילו להוסיף ולידציה שלילית (הפוכה), כלומר, הוסיפו תנאי שמחרוזת שגיאה נפוצה לא תופיע בתשובה של השרת.

כשאתם מתנסים עם Jmeter, תוכלו לפתוח את קובץ הדוגמא example.jmx מתוך הממשק של הכלי. כזכור, אין ממשק ל- K6. במקרה זה תריצו את הדוגמא example.js מתוך שורת הפקודה הבאה: K6 run example.js ואם תרצו לערוך את התסריט אז



בדיקת ביצועים ועומסים עם JMETER או K6 בעידן AI/ML | שי גינזבורג

בניגוד ל- Jmeter, עם K6 תצטרכו לפתוח כל עורך טקסטים, למשל Notepad.

את הנושא של ביטויים רגולריים REGEX ואת הטיפול הנדרש בפרמטרים דינמיים נשאיר למסגרת אחרת (למשל, המקרה שבו הלקוח מקבל מהשרת "Session ID" חד-פעמי חדש אחרי כל לוג-אין מוצלח בתוךResponse Header, והוא צריך להשתמש בו בתוך כל Request Header בכל המשך השימוש באפליקציה, אחרת השרת יזרוק אותו שוב לעמוד הכניסה). נציין רק, שבהעדר דפדפן (כמו שהסברתי קודם) נצטרך לטפל בכל אלמנט מהותי שבבדיקות אוטומטיות ברמת ממשק המשתמש של הדפדפן (GUI) טופל על ידי הדפדפן בעצמו. תוכלו להמשיך לשחק בדוגמאות שהכנתי עבורכם ולחפש למשל, היכן קובעים את מספר הגולשים הווירטואליים, את מספר מחזורי הבדיקה, את זמן ההמתנה בין דגימה לדגימה, וכל פרמטר אחר שמעניין אותכם להכיר. ככל שתתעמקו ותתנסו בשני הכלים, תשדרגו את המקצועיות שלכם ואת היכולות הטכניות שלכם.

חשוב מאד - במיוחד אם אתם מריצים את הכלים ואת הדוגמאות על מחשב פרטי ולא על שרת עוצמתי בענן, שימו לב לניצול המשאבים של המחשב אשר מפיק את העומס. אם המשאבים המקומיים של המחשב מוצו (למשל, CPU 100% עסוק, או צריכת זיכרון RAM מקסימלית, או שימוש ברשת ללא רוחב פס משמעותי) אז כל התוצאות של בדיקות העומס יהיו גרועות, אבל זה נובע רק מחוסר תיכנון המשאבים המקומיים ולא קשור לביצועים של האפליקציה בכלל כמובן.

אינטליגנציה מלאכותית ולמידת מכונה בהקשר של בדיקות עומסים וביצועים

בינה מלאכותית (AI) יכולה לשפר משמעותית את תהליך הכנת סקריפטים לבדיקת עומס של מכל סוג ועבור כל כלי על ידי שילוב ברמה שלא הייתה כמוה מעולם של אוטומציה, אופטימיזציה ותובנות חכמות. הנה כמה דרכים (אני בטוח שיש עוד הרבה דרכים אחרות) בהן ניתן למנף את החדירה של AI לתוך התחום:

פרמטריזציה דינמית עבור קלט בצד המשתמש: אלגוריתמי AI יכולים לזהות פרמטרים דינמיים (כגון Tokens, Session ID, חותמות זמן או מזהי משתמש) ולהחליף אותם אוטומטית בערכים רלוונטיים במהלך ביצוע הבדיקה. התהליך הזה יבטיח שסקריפטים יישארו מדויקים וניתנים להתאמה גם כשהאפליקציה משתנה, כשהתאריכים מתקדמים, כאשר נתוני המשתמשים מתחלפים ועוד.

מתאם (Correlation) וחילוץ נתונים (Extraction) מתוך פלט בצד השרת: בינה מלאכותית יכולה לזהות מתאמים (קורלציות) בין בקשות ותגובות, ועוזרת לבודקים לחלץ את הנתונים הדרושים (למשל, Verification Tokens, Cookies) עבור בקשות עוקבות. התהליך הזה יפחית את המאמץ הידני בזיהוי וטיפול בערכים דינמיים.

אופטימיזציה של פרמטרים: אלגוריתמים של בינה מלאכותית יכולים לבצע אופטימיזציה של פרמטרי בדיקה (כגון ספירת Threads, זמן Response וזמן Think time) בהתבסס על התנהגות המערכת, זמינות המשאבים ויעדי ביצועים. התהליך הזה יבטיח ניצול יעיל של משאבים וסימולציית עומס מדויקת.

חיזוי ושחזור שגיאות: מודלים של AI יכולים לחזות צווארי בקבוק פוטנציאליים, פסקי זמן או שגיאות במהלך בדיקות עומס. ניתן יהיה לשפר סקריפטים לבדיקה עם לוגיקה מותאמת אישית כדי לטפל בתרחישים כאלה.

תחזוקת סקריפט: AI יכול לעקוב אחר שינויים באפליקציה ולעדכן אוטומטית סקריפטים לבדיקה. הוא מזהה קישורים 'שבורים' (404), נקודות קצה (End Points) שהוצאו משימוש או פרמטרים שהשתנו.

ממשקי שפה טבעית: צ'אט בוטים המופעלים על ידי בינה מלאכותית או ממשקי שפה טבעית יכולים לסייע לבודקים ביצירת תרחישי בדיקה. בודקים יכולים לתאר את הדרישות שלהם בשפה פשוטה, ובינה מלאכותית מתרגמת אותן לתסריטי Jmeter או K6 או כל כלי אחר.

למידה התנהגותית: בינה מלאכותית יכולה ללמוד מתוצאות ביצוע בדיקות ולהתאים סקריפטים לאורך זמן. האלגוריתמים יזההו דפוסים, חריגות ורגרסיות ביצועים.

מסקנות לסיכום ובחירת הכלי הנכון לבדיקות עומסים וביצועים

לסיכום, החוזקות של JMeter נעוצות בריבוי ההרחבות שלו (כמעט כולן Free Open Source), בתמיכה בפרוטוקולים מגוונים (לא רק Http/s), ממשק ידידותי למשתמש (אפילו שנראה מיושן) וכלי דיווח חזקים (Offline Report & Analysis). בסופו של יום, זהו כלי אמין עבור אנשי מקצוע בתחום איכות התוכנה וספקי שירותי בדיקות ביצועים. הכלי מתאים לבדיקות של סביבה יחסית עתירת משאבים – הוא צורך יותר זיכרון הודות לארכיטקטורה מבוססת Java. השתמשו ב-JMeter כאשר החוזקות שלו משתלבות עם הצרכים שלכם, וכאשר אתם אנשי QA אשר צריכים להתחיל לבדוק במהירות ואז להפיק ממצאים כדי לקדם את הפרויקט. הכלי הוא אופטימלי למצבים שבהם נדרש לכם כלי בעל GUI. החוזקות של K6 הן סקריפטים ידידותיים למפתחים בשפת JavaScript, ביצוע מקבילי, יכולת הדמיה של תרחישים מציאותיים וגישה מודרנית לבדיקות מבחינת שילוב עם CI/CD. השתמשו ב-K6 כאשר אתם מעדיפים את חוויית המפתחים, למשל בבדיקות אימות הפיתוח, לפני המסירה לאנשי QA, ומעדיפים סקריפטים מבוססי קוד JavaScript. הכלי הזה אידיאלי לצוותים בעלי ידע טכנולוגי, אנשי פיתוח, ולמי שמחפש מדרגיות (Scalability). הכלי מתאים מאוד לבדיקות עומס של אפליקציות ומיקרו-שירותים מודרניים.

אנחנו נמצאים ברגע מיוחד בהיסטוריה של המחשוב. בקרוב יתחילו התוכנות לכתוב את עצמן והתעשייה של ההי טק תשנה את פניה באופן שבו קצב הפיתוח יהיה רובוטי, ולכן קצב הבדיקות יהיה חייב להדביק את הפער. יישום AI בבדיקות תוכנה הוא לא אופציה נחמדה, אלא הישרדות מקצועית. בכנס רפואי שנערך לא מזמן, אמר אחד הדוברים הראשיים, ש-AI לא יחליף את הרופאים, אבל רופאים שלא יעבדו עם AI הם אשר יוחלפו. עליכם לזכור, שבעוד ש-AI יכול לשפר את הכנת הסקריפט של Jmeter או K6 או כל כלי אחר, חיוני לאמת ולכוונן את הסקריפטים שנוצרו באופן ידני. בנקודת הזמן הנוכחית, המומחיות האנושית נותרה חיונית להבנת ההקשר העסקי, יעדי הבדיקה ותרחישים ספציפיים.

בדיקות קומבינטוריות
Combinatorial Testing

בעידן המודרני של פיתוח תוכנה, מורכבות המערכות עלתה באופן דרמטי. כתוצאה מכך, הבטחת האמינות והפונקציונליות של מערכות אלו הפכה למאתגרת יותר. אסטרטגיה יעילה אחת להתמודדות עם אתגר זה היא בדיקה קומבינטורית. גישת בדיקה זו מתמקדת ביצירה וביצוע שיטתי של מקרי בדיקה על מנת לכסות שילובים שונים של פרמטרי קלט, ומספקת הערכה יסודית של התנהגות המערכת בתנאים מגוונים.

בדיקה קומבינטורית ממנפת את הרעיון שתקלות תוכנה רבות מופעלות על ידי אינטראקציות של מספר קטן של פרמטרים. על ידי כיסוי כל השילובים האפשריים של פרמטרים אלה עד לרמה מסוימת (למשל, זוגות, שלשות), בדיקה קומבינטורית יכולה לזהות תקלות שעלולות להתפספס בשיטות בדיקה אחרות.

כמובן שכולנו זוכרים את אחד העקרונות החשובים בבדיקות שטוען שלא ניתן לבדוק את כל האפשרויות (Exhaustive Testing is Impossible) יחד עם זאת, שימוש בבדיקות אלו מאפשר לנו להתקרב לכיסוי מירבי ככל שניתן וזה בהחלט עדיף גם אם לא נגיע לכיסוי מלא של כלל המצבים.

טור זה מתעמק בעקרונות, בשיטות, ביתרונות ויישומים של בדיקה קומבינטורית, יחד עם האתגרים והסיכויים העתידיים שלה.

קובי יונסי

בעל תואר ראשון מאונ' בר אילן בתחום הלוגיסטיקה והכלכלה. מייסד את המרכז המוביל לבדיקות תוכנה, ספק ההדרכה הראשי בבתי תוכנה הגדולים בישראל. מנהל בוטקאמפים ותוכניות הכשרה לבודקי תוכנה. באקדמיה מרצה מוביל בטכניון, במכללה להנדסה עזריאלי י-ם, באקדמיה רמת גן ובמרכז האקדמי פרס רחובות. מלמד אנשים וארגונים כיצד לחשוב בדיקות ומוביל מדי שנה מאות בוגרים לתעשיית ההיי-טק.

עקרונות של בדיקה קומבינטורית

פיצוץ קומבינטורי

הבסיס של בדיקה קומבינטורית טמון בהבנת המושג פיצוץ קומבינטורי (באנגלית: Combinatorial Explosion). הכוונה היא לגידול המהיר במספר מקרי הבדיקה האפשריים ככל שמספר פרמטרי הקלט והערכים האפשריים שלהם גדלים. לדוגמה, למערכת עם 10 פרמטרי קלט בינאריים (כל אחד יכול להיות 0 או 1) תהיה 2 בחזקת 10 כלומר 1024 שילובים אפשריים.

בדיקה קומבינטורית שואפת לנהל את הפיצוץ הזה על ידי התמקדות בתת-קבוצה של שילובים שסביר להניח שהם אלו שיחשפו תקלות אפשריות.

כיסוי אינטראקציה

עיקרון מפתח בבדיקה קומבינטורית הוא כיסוי אינטראקציה. זה כולל הבטחה שכל האינטראקציות האפשריות של פרמטרי קלט נבדקות במידה מסוימת. הרמות הנפוצות ביותר של כיסוי אינטראקציות הן:

בדיקה זוגית (בדיקה דו-כיוונית): מבטיחה שכל הזוגות האפשריים של ערכי פרמטרי קלט מכוסים.

למשל כשנרצה לבדוק אימות זהות מבוסס על שם משתמש וסיסמא:

שם משתמש	סיסמא	תוצאה
תקין	תקינה	כניסה למערכת
תקין	לא תקינה	הודעת שגיאה
לא תקין	תקינה	הודעת שגיאה
לא תקין	לא תקינה	הודעת שגיאה

בדיקת t-way (כאשר t > 2): מרחיב את הרעיון לכיסוי כל השילובים האפשריים של פרמטרים t.

למשל כשנרצה לבדוק סיסמא תקינה/לא תקינה על פני משתמשים שונים:

שם משתמש	סיסמא	תוצאה
משתמש 1	תקינה	כניסה מאושרת
משתמש 1	לא תקינה	כניסה מסורבת
משתמש 2	תקינה	כניסה מאושרת
משתמש 2	לא תקינה	כניסה מסורבת
משתמש 3	תקינה	כניסה מאושרת
משתמש 3	לא תקינה	כניסה מסורבת

יצירת סוויטות בדיקה

תהליך הבדיקה הקומבינטורית כולל יצירת ערכת בדיקות המכסה ביעילות את רמת האינטראקציה הרצויה. זה מושג בדרך כלל באמצעות אלגוריתמים הבונים סט מינימלי של מקרי בדיקה תוך מיקסום הכיסוי. טכניקות כגון מערכי כיסוי ומערכי כיסוי ברמות מעורבות משמשות לעתים קרובות כדי להשיג מטרה זו.

כיצד נממש בדיקות אלו?

בכדי לבצע בדיקות אלו נצטרך לפעול בדרך הבאה:

1. יש לזהות תחילה פרמטרים של קלט ופלט וקומבינציות שלהם על פי אפיון המוצר (הגדרות משתנים, תכונות, אובייקטים וכו')
2. יש להגדיר עבור כל אחד מהפרמטרים רשימת אפשרויות שמייצגות הגדרות ומשתנים שונים (למשל: שמות משתמשים, סיסמאות, הגדרות תפקיד, פרופילים במערכת וכו')
3. הכנס את המשתנים השונים לכלי בדיקה וצור לעצמך מגוון של מקרי בדיקה אפשריים שכוללים קומבינציות שונות של משתנים.
4. לאחר שיצרת לעצמך את מגוון האפשרויות בטבלאות החלטה או רשימות של מקרי בדיקה אפשריים – הרץ את המקרים.
5. נתח את תוצאות הבדיקה וסקור את כיסוי הבדיקה שקיבלת ביחס לדרישות הלקוח כפי שמופיעות במפרט הטכני.
6. זהה ודווח על בעיות שנוצרו בעקבות מימוש הבדיקות הקומבניטוריות.
7. המשך להריץ את הבדיקות תוך תחזוקה של מקרי הבדיקה, שינויים ועדכונים שנחוצים תוך כדי תנועה.

היתרונות של בדיקה קומבינטורית

זיהוי תקלות טוב יותר

בדיקה קומבינטורית יעילה ביותר באיתור תקלות הנגרמות על ידי אינטראקציות בין פרמטרי קלט. מחקרים הראו שניתן לייחס חלק ניכר מהפגמים בתוכנה לאינטראקציות מסוג זה. על ידי כיסוי שיטתי של אינטראקציות

האנציקלופדיה לבדיקות | קובי יונסי

אלה, בדיקה קומבינטורית יכולה לגלות תקלות שכנראה נפספס בשיטות בדיקה אחרות.

הפחתת מאמצי הבדיקה

אחד היתרונות העיקריים של בדיקה קומבינטורית הוא היכולת שלה להפחית את מספר מקרי הבדיקה הדרושים להשגת כיסוי מקיף. זה מועיל במיוחד במערכות גדולות שבהן בדיקה ממצה אינה מעשית. על ידי התמקדות בשילובים קריטיים של פרמטרי קלט, בדיקה קומבינטורית מספקת גישה יעילה וחסכונית יותר לבדיקות תוכנה.

יישום על תחומים מגוונים

בדיקות קומבינטוריות הן צדדיות וניתן ליישם אותן במגוון רחב של תחומים, כולל תוכנה, חומרה, מערכות מדיקל, טלקומוניקציה, תעופה, חלל ומערכות צבאיות. העקרונות שלה ישימים על כל מערכת עם פרמטרי קלט ואינטראקציות מרובות, מה שהופך אותה לכלי בעל ערך בתעשיות שונות.

כיסוי מבחן משופר

על ידי הבטחה שכל השילובים הרלוונטיים של פרמטרי קלט נבדקים, בדיקה קומבינטורית משפרת את כיסוי הבדיקה הכולל. גישה יסודית זו עוזרת לזהות מקרי קצה ותנאי גבול, כאלו, שאם נשתמש בטכניקות בדיקה אחרות אנו עלולים להתעלם מהם, מה שמוביל למערכות יציבות ואמינות יותר.

חסרונות של בדיקה קומבינטורית

מורכבות מימוש

בדיקה קומבינטורית עשויה להיות מורכבת יותר כאשר מתמודדים עם מספר עצום של פרמטרים וערכים של קלט. זה יהיה קצת מאתגר עבור הבודקים לנהל את השילובים האלה באופן ידני.

שילוב אוטומציה מאתגר

כלי בדיקה קומבינטוריים אוטומטיים עשויים לדרוש יותר משאבי מה שיכול להפוך את הבדיקות למאתגרות עבור פרויקטי תוכנה שמתפקדים עם מספר קטן של משאבים..

חוסר הלימה בין הבדיקות למשתמשים

בדיקה קומבינטורית מתמקדת בשילובי קלט ספציפיים ולפעמים הם עשויים שלא לייצג התנהגות משתמשים בעולם האמיתי.

תחזוקה של מקרי בדיקה

עם השינויים שמתווספים לאפליקציית התוכנה המסוימת, יש לעדכן מקרי בדיקה קומבינטוריים ולתחזק היטב בעקבות שינויים בתוכנה מגירסא לגירסא.

"בדיקה קומבינטורית יעילה ביותר באיתור תקלות הנגרמות על ידי אינטראקציות בין פרמטרי קלט"

יישומים של בדיקה קומבינטורית

בדיקות תוכנה

בבדיקות תוכנה, בדיקות קומבינטוריות משמשות לאימות הפונקציונליות והביצועים של יישומים. זה יעיל במיוחד בבדיקת תצורות, קלט משתמשים ונקודות אינטגרציה. לדוגמה, בבדיקת יישומי אינטרנט, ניתן להשתמש בבדיקה קומבינטורית כדי להבטיח שכל השילובים האפשריים של סוגי דפדפן, מערכות הפעלה ופעולות משתמש נבדקים.

בדיקת חומרה

בדיקה קומבינטורית משמשת גם בבדיקות חומרה כדי לאמת את האינטראקציות בין רכיבים שונים. לדוגמה, בתכנון ובדיקה של מעגלים משולבים, בדיקה קומבינטורית מבטיחה שכל שילובי האותות האפשריים מוערכים, ועוזרת לזהות ולמגר תקלות.

תקשורת

בתעשיית הטלקומוניקציה, בדיקות קומבינטוריות משמשות לבדיקת יכולת הפעולה ההדדית של פרוטוקולי רשת והתקנים. על ידי כיסוי כל האינטראקציות האפשריות בין פרמטרים שונים של פרוטוקול, בדיקה קומבינטורית מסייעת להבטיח תקשורת והעברת נתונים אמינים בין רשתות.

תעופה, חלל ומערכות צבאיות

בתעשייה האווירית, בדיקות קומבינטוריות מיושמות כדי לבדוק את האינטראקציות בין תת-מערכות ורכיבים שונים. זה חיוני להבטחת הבטיחות והאמינות של מערכות תעופה וחלל, שבהן אפילו לתקלות קלות יכולות להיות השלכות קטסטרופליות.

אתגרים של בדיקה קומבינטורית

מדרגיות Scalability

אחד האתגרים העיקריים של בדיקות קומבינטוריות הוא מדרגיות. ככל שמספר פרמטרי הקלט והערכים האפשריים שלהם גדל, מספר השילובים האפשריים גדל באופן אקספוננציאלי. בעוד שטכניקות כמו OAs ו-CAs עוזרות לנהל את המורכבות הזו, יצירה וביצוע של מקרי בדיקה עבור מערכות גדולות מאוד עדיין יכולים להיות מאתגרים.

יצירת מקרה מבחן

יצירת סט אופטימלי של מקרי בדיקה המספק את הרמה הרצויה של כיסוי אינטראקציה היא משימה לא טריוויאלית. בעוד אלגוריתמים וכלים שונים זמינים, מציאת חבילת הבדיקות היעילה ביותר דורשת לעתים קרובות משאבי חישוב משמעותיים ומומחיות גבוהה, כזאת שניתן להשיג לעיתים רק על ידי יועצים חיצוניים.

ביצוע מבחן

ביצוע מקרי הבדיקה שנוצרו, במיוחד במערכות גדולות ומורכבות, עשוי להיות גוזל זמן ומשאבים. הבטחה שסביבת הבדיקה משקפת במדויק את התנאים בעולם האמיתי וניהול ביצוע של מספר רב של מקרי בדיקה הם אתגרים מעשיים שיש לטפל בהם.

אחד האתגרים העיקריים של בדיקות קומבינטוריות הוא מדרגיות. ככל שמספר פרמטרי הקלט והערכים האפשריים שלהם גדל, מספר השילובים האפשריים גדל באופן אקספוננציאלי"

תמיכה בכלים

אמנם ישנם מספר כלים זמינים לבדיקה קומבינטורית, אך היכולות וקלות השימוש שלהם עשויים להשתנות. בחירת הכלי המתאים לאפליקציה ספציפית ושילובו בתשתית הבדיקה הקיימת דורשת שיקול ותכנון מדוקדקים.

עתידם של בדיקות קומבינטורית

התקדמות באלגוריתמים

מחקר מתמשך בבדיקות קומבינטוריות מתמקד בפיתוח אלגוריתמים יעילים יותר ליצירת מקרי בדיקה. ההתקדמות בלמידת מכונה ובינה מלאכותית טומנת בחובה הבטחה לאוטומציה של יצירת חבילות בדיקה אופטימליות ושיפור השימוש בסוג בדיקות אלו.

התממשקות עם שיטות בדיקה אחרות

התפתחויות עתידיות בבדיקות קומבינטוריות עשויות להיות כרוכות בהתממשקות הדוקה יותר עם שיטות בדיקה אחרות, כגון בדיקות מבוססות מודל, בדיקת מבוססות התנהגות, בדיקות מונחות סיכון ועוד. גישה משולבת זו יכולה לספק אסטרטגיית בדיקה מקיפה יותר, תוך מינוף החוזקות של כל שיטה.

האנציקלופדיה לבדיקות | קובי יונסי

תמיכה בכלים משופרת

ככל שבדיקות קומבינטוריות ממשיכות לצבור כוח, אנו יכולים לצפות לראות תמיכה משופרת בכלים עם ממשקים ידידותיים יותר למשתמש ותכונות מתקדמות.

יישום לטכנולוגיות מתפתחות

בדיקות קומבינטוריות ימלאו תפקיד חשוב יותר ויותר בבדיקת טכנולוגיות מתפתחות כמו האינטרנט של הדברים (IoT), מערכות אוטונומיות ובינה מלאכותית. טכנולוגיות אלו כוללות אינטראקציות מורכבות בין רכיבים רבים, מה שהופך את הבדיקה הקומבינטורית לגישה אידיאלית להבטחת המהימנות והפונקציונליות שלהם.

מסקנות

בדיקה קומבינטורית היא גישת בדיקה רבת עוצמה ורב-תכליתית הנותנת מענה לאתגרים של הבטחת האמינות והפונקציונליות של מערכות מורכבות. על ידי כיסוי שיטתי של אינטראקציות בין פרמטרי קלט, בדיקה קומבינטורית משפרת את זיהוי התקלות, מפחיתה את מאמץ הבדיקה ומשפרת את כיסוי הבדיקה. אומנם נותרו אתגרים בשיטת הבדיקה הזאת אבל, מחקר מתמשך והתקדמות באלגוריתמים ותמיכה בכלי בדיקה יסייעו לטפל בבעיות אלו.

לסיכום

ככל שהמערכות ממשיכות לגדול במורכבותן, בדיקות קומבינטוריות יישארו כלי חיוני בארסנל בדיקות התוכנה והחומרה, כזה שיתרום לפיתוח מערכות חזקות ואמינות בתחומים מגוונים.

בחן את עצמך | ירון צוברי

הפתרון לשאלה

נתחיל עם יעד הלימוד (Learning Objective) ורמת הידע (Knowledge Level) הדרושים לנושא הזה. לידיעה כללית: את יעדי הלימוד ניתן לרוב למצוא בעמוד הפתיחה של כל פרק בתוכנית הלימוד (הסילבוס). לכל יעד לימוד מוגדרת רמת הידע הדרושה לו (K-level – Knowledge Level). המודל המייצג לרמות הידע שארגון ISTQB משתמש בו הוא מודל הטקסונומיה של בלום (Bloom's taxonomy).

יעד הלימוד הרלוונטי לשאלה שלנו כאן הוא: FL-5.6.1 - "כתוב דו"ח פגמים שמכסה פגם שנמצא במהלך הבדיקות"; והשאלה שלנו היא ברמה של K3. רמה זו אומרת שעל הנבחן להצביע על התשובה הנכונה מקום ויכולת של יישום ו/או שימוש של הנושא המסוים (או הנלמד במידה וניגש לבחינה לאחר הכשרה).

כאמור, נושא דיווח פגמים ותקלות נחשב לקשה ולרגיש ביותר, בעיקר כיוון שברגע שאנחנו מדווחים על תקלה, אנחנו נוגעים בנימי היצירה של המפתחים. לכן רצוי שנגלה מקצועיות בדיווח ונבצע זאת ברגישות. ככול שנהיה יותר מדויקים בפרטים, צמודים לעובדות, לשרשרת ההתרחשויות שהובילה לפגם, ולסימוכין שקל למפתחים להבין ולהשתמש בהן לטובת האנליזה של שורש התקלה, כך נוביל ליעילות העבודה על הפגם ובד בבד נפגין מקצועיות שכל מפתח תוכנה יעריך. חשוב להזכיר את מטרות לדיווחים על פגמים, ועיקרן:

- לספק למפתחים ולאחרים מידע על כל אירוע שלילי שהתרחש, לאפשר להם לזהות תוצאות ספציפיות, לבודד את הבעיה בעזרת בדיקת שחזור מינימלית ולתקן את הפגם האפשרי במידת האפשר או לפתור את הבעיה בכל דרך שהיא.
- לספק למנהלי בדיקות אמצעים למעקב אחרי איכות התוצרים ואחרי ההשפעה על הבדיקות (לדוגמה, אם מדווחים פגמים רבים, הבודקים ישקיעו זמן רב על דיווח במקום להריץ בדיקות נוספות ויהיה צורך ביותר בדיקות אימות).
- לספק רעיונות לשיפור תהליכי פיתוח ובדיקות.

על מנת לענות על המטרות הללו, נכתבה רשימה של מספר אלמנטים שרצוי שיהיו בתיעוד בכל דוח של פגם טיפוסי (לכזה שלרוב נמצא בבדיקות הדינמיות). לדוגמא (רשימה חלקית):

- מזהה
- כותרת/סיכום (Summary) ותיאור תמציתי קצר של הפגם המדווח
- תאריך הדיווח, הארגון שמדווח ומחבר הדוח
- זיהוי מושא הבדיקות (פריט התצורה הנבדק) והסביבה בה התגלה הפגם
- שלב מחזור חיי הפיתוח שבו התגלה הפגם
- וכו'

(למידע נוסף, מומלץ לקרוא את פרק 5 תת-פרק 5.6.1 של תוכנית ההכשרה הבסיסית של ארגון).

הבה נבחן את השאלה והתשובות:

השאלה שלנו עוסקת בעיקר בסיכום (summary) של התקלה ובאיכות של הסיכום שבאה לידי ביטוי בתפיסת מהות הכשל ואת מידת השפעתו על בעלי העניין. סיכום טוב של התקלה לעיתים מסביר את מהות התקלה ללא צורך אפילו להיכנס לפרטי התקלה, או לשלבי שחזור התקלה, או לקבצים נלווים וכיו"ב.

נסתכל עתה על התשובות השונות ונתחו האם הן נכונות או לא, ומדוע לא:

א. **התשובה אינה נכונה**. אמנם מידע זה שימושי למפתחים, אך אינו מספק למנהלים ו/או לבעלי העניין את התחושה של ההשפעה על איכות המוצר.

ב. **התשובה אינה נכונה**. סיכום זה אינו מספק למפתחים או למנהלים ו/או לבעלי העניין את המידע הדרוש ויתרה מזאת אף תוקף את המפתחים...

ג. **התשובה אינה נכונה**. סיכום זה אינו מספק למפתחים או למנהלים ו/או לבעלי העניין את המידע הדרוש ויתרה מזאת, אף כאן ישנה התקפה כנגד המפתחים...

ד. **התשובה <u>נכונה</u>**. סיכום זה נותן תחושה טובה של הבנת הכשל ומידת השפעתו.

אם תרצו שאציג שאלת דוגמה בנושא מסוים או אם יש לכם שאלות אנא פנו אלי באימייל: yaron.tsubery@itcb.org.il

בחירת שפת תכנות עבור תשתית אוטומציה שיקולים, אילוצים ועוד ניסים אריאל

ניסים אריאל

ראש צוות בדיקות בחברת X-trodes, מספר שנים בתחום הבדיקות, נשוי+6 וחובב היסטוריה ופילוסופיה

מאמר זה אינו בא לתת פתרון לכל המקרים ולכל המוצרים, אלא לתת מבט רוחבי וכללי על מנת לקבל מושג מעולם האוטומציה לפני שקופצים אליו. בנוף הטכנולוגי המתפתח במהירות רבה בימינו, אוטומציה למוצר היא כלי בדיקות **הכרחי** לייעול תהליכים, כמו בדיקות רגרסיה ארוכות שמתקצרות, והגדלת היעילות של הבודקים על ידי פינוי זמן למשימות חשובות. הפחתת עומס זה מייצרת תועלת רבה בצורת פיתוח ובדיקות תוכנה נוספים. החלטה מרכזית ביותר בפיתוח תשתית אוטומציה היא בחירת שפת התכנות המתאימה. לעומת המצב לפני 7-6 שנים, כיום ישנו שפע של אפשרויות זמינות, שפות שונות על גבי פריימוורקים שונים ולכל אחת החוזקות והחולשות שלה. בחירה נכונה יכולה להשפיע באופן משמעותי על הצלחת הפרויקט. במאמר זה, נרחיב מעט בשיקולים ובגורמים המנחים בבחירת שפת התכנות האופטימאלית עבור מסגרת האוטומציה הרצויה.

השפעות חוץ

פעמים רבות מנהל הפיתוח כבר בשלב הגיוס מכוון לשפה מסוימת. מניסיוני, זו השפה העיקרית שבה מפותח המוצר בחברה ובה נכתבים Unit tests. באופן כמעט טבעי, מנהל המוצר רוצה להמשיך עם הסטאק הטכנולוגי שהוא מחזיק, ואפילו יש לו מחשבה ליצור אינטגרציה בין האוטומציה העתידה לקום לבין מה שהפיתוח כותב.

מחשבה זו אינה בטלה ולעיתים היא נכונה וזה מאוד תלוי מקרה, אך היא אינה לוקחת בחשבון כמה נקודות קריטיות:

- הרוב המוחלט של פלטפורמות האוטומציה נכתבות כ Standalone, כך שהשפה שבה מפותח המוצר איננה כ"כ רלוונטית בשיקול הבחירה.
- לא תמיד השפה שבה המוצר מפותח מתאימה לאוטומציה, למשל C++ שהיא שפה נפוצה מאוד בעולם ה-Embedded, אך לא בנויה לשימושים של אוטומציה קלאסית של UI או בדיקות API. אפשר להקים איתה מערכי בדיקות למוצרי Embedded, אבל בצורה שיותר מתאימה לעולם הפיתוח מאשר לעולם הבדיקות.
- האם השפה שמכוונים אליה היא שפה עם אפשרויות, עם ספריות שנכתבו לטובת אוטומציה. הרי כתיבת אוטומציה, היא בעלת אופי ייחודי ששונה מפיתוח בשורה התחתונה, ובדרך כלל על הבסיס הזה נוצרו ספריות אוטומציה לשפה מסוימת ולאחרת הרבה פחות. לדוגמא: PYTHON – שיש לה תאימות להרבה פלטפורמות, לעומת NET. שלא.
- הכשרת צוות – כמובן שאם נצטרך להכשיר צוות שלא כולו מגיע מרקע של פיתוח, נרצה לעבוד בשפה שהיא High level, ולא להעביר לצוות קורס שלם במדמ"ח.

הבנת הצרכים והדרישות

לפני הצלילה לתהליך הבחירה, חשוב להבין היטב את הצרכים של הפרויקט. האם התשתית היא עבור מוצר WEB, או שמא Mobile, או מוצר שהוא Desktop. כל אלה מאוד חשובים בעיקר בגלל ההיצע המשתנה בהתאם לסוג המוצר. ישנם מוצרים מתחום ה-Embedded שאין פלטפורמות של אוטומציה שנבנו מתוך מחשבה עליהם, בדרך כלל משום שהם עלולים להיות ייחודיים, מה שמקטין באופן דרסטי את הדרישה והצורך בשוק לפיתוח תשתית כזו. תשתיות עבור מוצרי Web נפוצים ביותר, פחות מזה בתחום ה-Mobile, ואף פחות בתחום ה-Desktop applications.

ובכן מה עושים אם מגלים שהמוצר שלנו לא יכול להיבדק באופן שלם על ידי תשתית אוטומציה כלשהיא? למשל מוצרים מהתחום הרפואי, או הביטחוני...במקרה כזה אין ברירה אלא להגדיל ראש ולקפוץ למים העמוקים של השפה שבה המוצר מתפתח ולהיעזר בפיתוח על מנת ליצור כיוון לכתיבת בדיקות בקוד, וכמו כן איך לתכנן את התשתית כדבר שניתן לתחזק ולהשתמש בו באופן סדיר. כמובן שמדובר באתגר לא פשוט כלל, שחברות רבות מפקידות אותו בידי הפיתוח.

קהילה

נושא סופר חשוב. אמנם יש מקרים כמתואר לעיל שקופצים למים עמוקים ועלולים למצוא את עצמינו כותבים אוטומציה עם Kotlin ו-Gradle אך זהו ממש מקרה קצה. בסופו של דבר איננו רוצים ואין זה חכם להתיימר להמציא לבד את הגלגל מחדש. קהילה שתומכת יכולה להוות גורם ממנף בהיתקלות בכל מיני בעיות ייחודיות או מקרים שונים, תוך כדי הקמת תשתית או תחזוקה שלה. לכן זה שיקול ממשי שיש להכניס לסל הפרמטרים בהחלטה.

הערכת התאמת שפה

לשפות תכנות שונות יש מאפיינים שונים ההופכים אותן למתאימות למשימות ספציפיות. עבור מסגרות אוטומציה, גורמים כמו קריאה, תחזוקה, ביצועים ותמיכה בקהילה. להלן כמה שפות תכנות פופולריות והתאמתן לאוטומציה:

מספר דוגמאות עם Pross and Cons

Python

Python היא שפת תכנות high-level, רב-תכליתית ובעלת כתיבה דינמית, הידועה בפשטות ובקריאות שלה. כוללת תחביר ברור המדגיש את נוחות הקוד ומפחית את עלות תחזוקת התוכנית. Python תומכת במספר פרדיגמות תכנות, כולל תכנות פרוצדורלי, מונחה עצמים ופונקציונלי, מה שהופך אותה למתאים למגוון רחב של יישומים כגון פיתוח אתרים, ניתוח נתונים, בינה מלאכותית, מחשוב מדעי, אוטומציה ועוד. הספרייה הסטנדרטית הנרחבת שלה והמערכת האקולוגית העצומה של ספריות ומסגרות של צד שלישי תורמים לפופולריות שלה בקרב מפתחים עבור מתכנתים מתחילים ומנוסים כאחד.

> "החלטה מרכזית ביותר בפיתוח תשתית אוטומציה היא בחירת שפת התכנות המתאימה"

יתרונות

1. **פשטות התחביר**. Python הוא פשוט וקריא, מה שהופך אותו לנגיש בכל רמות המיומנות. פשטות זו מאיצה את תהליך הפיתוח ומקדמת תחזוקה של קוד.
2. Python מתהדרת במערכת אקולוגית עשירה של **ספריות** למשימות אוטומציה. ספריות פופולריות כמו Selenium, PyTest מספקות כלים חזקים לבדיקות אינטרנט, בדיקות API ועוד.
3. **תמיכה בקהילה**. לפייתון יש קהילה משגשגת של מפתחים שתורמים לצמיחתה ומספקים תמיכה באמצעות פורומים, תיעוד ופרויקטים בקוד פתוח.

חסרונות

1. **ביצועים.** בעוד ש-Python מצטיינת בתחביר נוח, ייתכן שהיא לא הבחירה הטובה ביותר מבחינת מהירות ביצועים. האופי הדינאמי של Python בצורת כתיבתו יכול להוביל למהירויות ביצוע איטיות יותר בהשוואה לשפות עם הקלדה סטטית כמו Java או C#.
2. **ניהול תלויות.** ניהול תלות ומעטפת של יישומי Python יכול להיות מאתגר, במיוחד במסגרות אוטומציה מורכבות עם מספר רב של ספריות חיצוניות. עלולים להתעורר התנגשויות, בעיות ניהול גרסאות וחששות תאימות, הדורשים ניהול זהיר.
3. **עקומת למידה.** למרות הפשטות, שליטה ב-Python לאוטומציה דורשת זמן ומאמץ, במיוחד עבור מפתחים ללא ניסיון קודם עם השפה. בעוד שקלות הקריאה של Python עשויה להוריד את מחסום הכניסה, לימוד הניואנסים הדקים והשיטות המומלצות לפיתוח אוטומציה עדיין יכולה להיות משימה משמעותית.

פלטפורמות נפוצות ש-Python עובדת איתן:
Selenium, Appium, Playwright, Robot Framework, Cucumber

בחירת שפת תכנות עבור תשתית אוטומציה שיקולים, אילוצים ועוד ניסים אריאל

Java

Java היא שפת תכנות בשימוש נרחב, מונחה עצמים, בלתי תלויה בפלטפורמה, הידועה בניידות וברבגוניות שלה. זה מאפשר למפתחים לכתוב קוד פעם אחת ולהפעיל אותו בכל פלטפורמה שתומכת ב-Java. משמשת לעתים קרובות לבניית יישומי אינטרנט, אפליקציות סלולריות (אנדרואיד בעיקר), מערכות ארגוניות ויישומים בקנה מידה גדול בשל היציבות והרובסטיות שלה, ותכונות האבטחה שלה. ספרייה סטנדרטית עצומה, תמיכה קהילתית חזקה ומערכת אקולוגית עשירה של מסגרות וכלים.

יתרונות

1. **רובסטיות ואמינות.** הכתיבה הסטטית של Java, בדיקות המהדר הבולטות מיידית לעין ומנגנוני טיפול מקיפים בשגיאות תורמים לחוסן ולאמינותה של השפה. זה הופך את Java למתאים היטב לבניית מסגרות אוטומציה ניתנות להרחבה ויציבות.
2. **מערכת אקולוגית עשירה.** Java נהנית ממערכת אקולוגית עצומה של ספריות, כלים ומסגרות הנותנות מענה לצרכי אוטומציה שונים. בין אם מדובר באינטראקציה עם מסדי נתונים, טיפול בשירותי אינטרנט, JUnit, TestNG מציעות תמיכה נרחבת באמצעות ספריות וממשקי API של צד שלישי.
3. **קהילה וותיקה ביותר.** אין סוף חומר ודוגמאות ברשת. הדגמות, פתרון תקלות וכל מה שאפשר להעלות על הדעת.

חסרונות

1. **מורכבות ועקומת למידה.** האופי המונחה עצמים של Java והמערכת האקולוגית המקיפה של ג'אווה עשויים להציג עקומת למידה תלולה במיוחד אלה החדשים בשפה. שליטה ב-Java עבור אוטומציה דורשת הבנת מושגים כמו מחלקות, ירושה וממשקים, כמו גם היכרות עם כלים ומסגרות קשורות.
2. **בעיות סינכרוניזציה.** עלולות להיווצר עכב הכובד המשמעותי של השפה, החזקה ושחרור של זיכרון ומנגנון ה-Garbage collection שעלול להוות סוג של מעמסה על המטלות עלול ליצור פיספוס של אלמנטים ב-DOM.
3. **מסורבלות.** שפות יותר חדשות חותרות לכתיבה יותר מפושטת וקלה, ב-Java לעומת זאת היתרון הזה לא כל כך ניכר.

פלטפורמות נפוצות ש-Java עובדת איתן:
Selenium, Appium, Cucumber

JavaScript

JavaScript היא שפת תכנות מגוונת, קלת משקל וקריאה, המשמשת בעיקר ליצירת תוכן אינטראקטיבי ודינאמי באתרי אינטרנט. משמשת בעיקר בצד הלקוח (בדפדפן האינטרנט של המשתמש) והיא חיונית לפיתוח WEB. JavaScript מאפשר פונקציונליות, אנימציות, ואינטראקטיביות. עם עלייתו של Node.js, ניתן להשתמש ב-JavaScript גם בצד השרת, מה שמאפשר פיתוח full stack בשפה אחת. האימוץ הנרחב שלה, הספריות הנרחבות (כמו React, Angular ו-Vue.js), והשילוב הקל עם HTML ו-CSS הופכים אותו לכלי בסיסי עבור מפתחי אתרים.

יתרונות

1. **תפוצה רחבה.** JavaScript היא אחת משפות התכנות הנפוצות ביותר, במיוחד בפיתוח אתרים. הימצאותה כמעט בכל מקום מבטיחה שמפתחים רבים כבר מכירים את JavaScript, מה שמקל על ההכשרה של חברי הצוות ולמנף את המומחיות הקיימת כבר בחברה לפרויקטים של אוטומציה.
2. **אסינכרוניות.** מודל התכנות האסינכרוני – מקבילי, שאינו ממתין בתור, של JavaScript, המופעל על ידי תכונות כמו Promises ו-async/await, הופך אותו למתאים היטב לטיפול במשימות אסינכרוניות הנפוצות באוטומציה, כגון ביצוע בקשות HTTP, המתנה לאלמנטים שיופיעו ב-דף אינטרנט, וטיפול במשימות בעלות זמן קצוב ב-UI.
3. **אינטגרציה עם Node.js.** האינטגרציה של JavaScript עם Node.js מרחיבה את היכולות שלה מעבר לדפדפן, ומאפשרת למפתחים לבנות סקריפטים של אוטומציה בצד השרת ולמנף מגוון עצום של מודולי (Node Package Manager) npm - עבור אוטומציה מרובת משימות, כולל פעולות עם קבצים, רשתות ואינטראקציות עם מסד נתונים.

חסרונות

1. **כתיבה דינאמית.** הכתיבה הדינמית של JavaScript עלולה להוביל לשגיאות בזמן ריצה שאולי לא ייתפסו עד לביצוע, מה שמגביר את הסיכון לבאגים ולהתנהגות בלתי צפויה בסקריפטים. חוסר זה של בדיקת סוגים סטטיים יכול להפוך את תחזוקת הקוד ועיבוד מחדש למאתגרים יותר, במיוחד בבסיסי קוד גדולים. אם כי גם לזה כבר יש תוספים שיכולים לתת פתרונות.
2. **תאימות דפדפנים.** מאז ES6 – הגרסה האחרונה והמשמעותית ביותר של השפה, ישנן בעיות תאימות הדפדפן. הבדלים ביישום הדפדפן ובשונויות של מנוע JavaScript עשויים לחייב מאמץ נוסף כדי להבטיח תאימות בין דפדפנים והתנהגות עקבית בסביבות שונות.
3. אמנם האופי האסינכרוני של JavaScript הוא יתרון, אבל זה יכול גם להוביל **למבני קוד מורכבים ומקוננים,** המכונה בדרך כלל "Callback Hell" או "Promise chaining".

פלטפורמות נפוצות ש-JavaScript עובדת איתן:
Selenium, Cypress, Nightwatch.js, Puppeteer, .Playwright

TypeScript

TypeScript הוא ערכת-על של JavaScript שמוסיפה הקלדה סטטית אופציונלית ותכונות אחרות לשפה. מטרתה היא להרחיב את פיתוח JavaScript, לשפר תחזוקה ועמידות בפני שגיאות על ידי מתן אפשרות למפתחים להגדיר סוגים של משתנים, פרמטרים של פונקציות וערכי החזרה. מציעה קריאת קוד משופרת, תמיכה טובה יותר בכלים ובדיקת שגיאות משופרת, מה שהופך אותה לבחירה טובה לבניית יישומים בקנה מידה גדול, ושיפור פרודוקטיביות המפתחים.

יתרונות

1. שלא כאימה הבוגרת יותר JS - מספקת הקלדה סטטית, המאפשרת לתפוס שגיאות בזמן הידור ולא בזמן ריצה. דבר שמועיל בזיהוי כשלים פוטנציאליים בכתיבה בשלב מוקדם של תהליך הפיתוח, מה שמוביל לבדיקות חזקות ואמינות יותר.
2. תסריטי בדיקה שנכתבו ב-TypeScript קלים יותר לקריאה ולהבנה, במיוחד עבור מפתחים שחדשים בבסיס הקוד או בפרויקט. דבר שאני שומע מחברים שעברו מ-JS ל-TS.
3. ארגון ומבנה קוד טובים יותר. תכונות כמו ממשקים, מחלקות ומודולים עוזרות ביצירת קוד בדיקה נקי יותר וניתן לתחזוקה יותר.

חסרונות

1. לחלק מכלי אוטומציה מיוחדים או ספריות עשויות להיות תמיכה מוגבלת ב-TypeScript. זה עלול להוביל לאתגרים בשילוב כלים מסוימים במסגרת אוטומציה מבוססת.
2. המרת בסיסי קוד JavaScript קיימים ל-TypeScript יכולה לגזול זמן ועלולה לשגיאות. זה דורש התייחסות מדוקדקת של סוגים, ממשקים ושינויים פוטנציאליים.
3. גרסאות TypeScript צריכות להתיישר עם הגרסאות של ספריות ומסגרות אחרות בפרויקט. בעיות אי תאימות בין TypeScript לספריות של צד שלישי עלולות להתעורר ולדרוש מאמץ נוסף.

פלטפורמות נפוצות ש TypeScript עובדת איתן:
Cypress, Nightwatch.js, Puppeteer, Playwright

לסיכום:

בחירת שפת תכנות עבור מסגרת האוטומציה היא החלטה קריטית הדורשת התייחסות מדוקדקת לגורמים שונים, דרישות הפרויקט, התאמת השפה, מומחיות הצוות, כלי עבודה ותמיכה בקהילה. אמנם אין פתרון אחד שמתאים לכל המצבים והמוצרים, אבל Python, Java ו-JavaScript לדוגמא הן בחירות פופולריות עם מגוון רחב של פלטפורמות לשימוש, כל אחת עם נקודות החוזקה והחולשה שלה (שזה עוד נושא בפני עצמו). על ידי הערכה יסודית של גורמים אלה והתאמתם ליעדי הפרויקט, יש לקבל החלטה מושכלת המציבה את הבסיס להתפתחות של תשתית אוטומציה מועילה ומוצלחת.
בהצלחה!

עוד על פסיכולוגיה של בדיקות

כמו (כמעט) כל מי שכתב קוד, ולו רק כתרגיל בקורס, גם לי יצא לשרוף כמה שעות בניסיון להבין למה קוד שכתבתי לא עובד. במקרה שלי זה היה סוגריים לא סגורות – דבר שכלי הפיתוח הנוכחיים היו מגלים לי ברגע. אצל אחרים זו שורה של if (x = 1) במקום if (x == 1), או משהו ברמה הזאת. אחרי שעתיים של דיבאג ותלישת שערות, אתם חולקים את התסכול עם חבר. הנ"ל מעיף מבט אחד במסך, מצביע לכם על השטות ומשאיר אתכם תוהים איך לכל הרוחות לא ראיתם את זה בעצמכם.

אז יש סיבה טובה למה פספסתם את הברור מאליו: הדרך שבה המוח שלנו עובד. אני לא מתיימר להיות פסיכולוג או מומחה לקוגניציה, אבל כן חוויתי – וקצת קראתי – על ה"משחקים" שהמוח שלנו משחק איתנו, ואיך כל זה משפיע על איכות הבדיקות.

בגיליון 35 של המגזין התפרסם מאמר מעניין של **משי מוטולה** על הפסיכולוגיה של הבדיקות. המיקוד במאמר היה על היחסים והדרכים לתקשורת קונסטרוקטיבית בין בודקים למפתחים. במידה רבה זה גם המיקוד של הסעיף העוסק בנושא זה בסילבוס הבסיס של ISTQB. (הערה: הסעיף הזה "ירד במיקוד"... הוא היה קיים בוורסיה 3.1 של הסילבוס אבל קוצץ לטובת נושאים אחרים בוורסיה 4.0 ששוחררה בשנת 2023).

אספקטים אחרים של הפסיכולוגיה של בדיקות מוזכרים בסילבוס במספר מילים בלבד. במאמר זה אני רוצה להתמקד דווקא בנקודות אלה, שלדעתי חשוב מאוד להיות מודע להן – גם לבודקים מן השורה וגם למנהלי בדיקות.

מה המטרה?

יתכן מאוד שהדרך שבה נגדיר את מטרת הבדיקות תשפיע על כמות הבאגים שנזהה. בספר המצוין של Testing Computer Software (Kaner, Nugyen, Falk) מתואר הניסוי הבא (הציטוט בהשמטות): "צפו במסך מכ"ם וחפשו בליפ מסוים. דווחו על הבליפ בכל פעם שאתם רואים אותו [...] אם אתם מצפים לראות בליפים רבים, או אם תקבלו פרס גדול על דיווח על בליפים כשאתם רואים אותם, אתם תראו ותדווחו עליהם יותר - כולל בליפים שלא היו שם כלל ("אזעקות שווא"). אם אתם מאמינים שלא יהיו הרבה בליפים, או אם תענשו על אזעקות שווא, תפספסו בליפים שאכן הופיעו על המסך ("החמצות"). נדרשו לפסיכולוגים ניסויים כ-80 שנות ניסיון מר כדי להפסיק להאשים את נבדקי הניסוי בטעויות בניסויים מסוג זה ולהבין שלגישה ולהגדרת הניסוי [...] הייתה השפעה גדולה על הפרופורציות של אזעקות שווא והחמצות."

המסקנה של המחברים: "אם אתם מצפים למצוא באגים רבים, ואתם מקבלים שבחים או מתוגמלים על מציאתם, תמצאו הרבה. וכן - כמה מהם יהיו אזעקות שווא. אם אתם מצפים שהתוכנה תעבוד כהלכה, או אם אנשים מתלוננים כשאתם מוצאים בעיות ומענישים אתכם על אזעקות שווא, תחמיצו הרבה בעיות אמיתיות."

באחד הספרים הראשונים שנכתבו על המקצוע (The Art of Software Testing, Glenford Meyers, 1979), המחבר מסביר שאם אתם חושבים שהמשימה שלכם היא למצוא בעיות, אתם תחפשו אותן יותר מאשר אם אתם חושבים שהמשימה שלכם היא לוודא שאין בעיות. הוא מציע את ההגדרה הבאה: "בדיקות הוא תהליך של הרצת תוכנה מתוך מטרה למצוא בה טעויות".

כבודקים וכמנהלי בדיקות, אם נגדיר שזו המטרה, נהיה כנראה יותר אפקטיביים במציאת באגים.

ידע וציפיות משפיעים על פירוש התוצאות

כחלק מצוות הפיתוח, אנחנו יודעים די הרבה על ה"בפנוכו" של המוצר. רוב הסיכויים שיש לנו מושג על הארכיטקטורה, על איך יכולות מומשו, וגם הבנה למה יש למוצר מגבלות מסוימות. הידע הזה מונע מאיתנו לעיתים לזהות באגים. סיפור לדוגמה: לפני שנים רבות הייתי חבר בצוות שבדק דרייברים של כרטיסי Wi-Fi. יום אחד קיבלנו גרסה חדשה, ובדיקה מסוימת שבעבר עברה בהצלחה, נפלה. היה צורך לדבג את הנפילה על מנת לספק מידע למפתחים. חלק מתהליך הדיבוג היה הקלטה של התעבורה ברשת האלחוטית, שבעזרתה יכולים המפתחים לחפש מה גרם לתקלה. כיוון שבמעבדה פעלו עשרות כרטיסי Wi-Fi ותחנות גישה (access point), היה קשה מאוד לקבל הקלטה נקייה של התעבורה רק עבור המערכת הנבדקת. הפתרון היה כלוב פאראדיי בגודל3X3 מטר שהותקן במעבדה. כלוב זה חוסם כניסת גלים אלקטרומגנטיים, כך שאפשר היה לחבר מערכת שכללה רק כרטיס אחד ונקודת גישה אחת ולבצע הקלטה נקייה של התעבורה. אלא שנוצרה בעיה חדשה... כשהרצנו את הבדיקה בתוך הכלוב, היא עברה! הממם.... מה זה יכול להיות? המסקנה שלנו הייתה שכיוון שבמעבדה יש מלא משדרים ושליחה של תשדורות רבות, נוצרות התנגשויות בין השידורים, הם מפריעים אחד לשני, ומכאן הכשל (זה אכן יכול לקרות בתשדורות רדיו).

"אם אתם מצפים שהתוכנה תעבוד כהלכה... תפספסו הרבה בעיות אמיתיות"

האירוע חזר על עצמו גם בגרסאות הבאות: במעבדה: נכשל. בכלוב: עובר.

רק אחרי כחודש, מישהו החליט שההסבר על "רעש תשדורות" לא ממש מספק, נכנס ממש לעובי הקורה, וגילה שיש בעיה אמיתית בקוד שגורמת לכישלון בסביבות בהן יש הרבה נקודות גישה. שום קשר לכמות התשדורות באוויר...

את הנטייה הזאת, למצוא הסבר כמו-הגיוני לבעיות מסוימות מתוך (כביכול) הבנת המערכת, מסכם יפה דונלד נורמן בספרו The Design of Everyday Things: "מצאנו הסבר, ואנחנו מבסוטים. [...] ברגע שיש לנו הסבר – נכון או לא – עבור אירועים תמוהים, אין לנו יותר תהיות ואין אי התאמה בין הציפיות והמציאות. כתוצאה מכך, אנו שאננים, לפחות לזמן מה".

ידע משפיע על מה נראה לנו סביר

חלק מתפקידנו כבודקים הוא לייצג את המשתמשים. מהצד הפשוט והמכני זה אומר לבדוק אם המוצר ממלא את ההבטחות שלו מבחינה פונקציונאלית ומבחינת אספקטים של איכות (מהירות תגובה, למשל). אבל יש גם הגדרה לא מאוד מדויקת של

מיכאל שטאל

ארכיטקט בדיקות תוכנה באינטל, ישראל עוסק בעיקר בבדיקות מערכות משובצות מחשב. במסגרת תפקידו, מיכאל מגדיר שיטות בדיקה ומתודולוגיות עבודה, עוסק בהדרכה ולפעמים אפילו מרשים לו לבדוק משהו (שזה הכי כייף). מיכאל מציג תכופות בכנסים בארץ ובחו"ל ומלמד בדיקות תוכנה בפקולטה למדעי המחשב באוניברסיטה העברית. ניתן לראות חלק מהמצגות והמאמרים שלו באתר www.testprincipia.com

מחפש צרות | מיכאל שטאל

כמה נוח להשתמש במוצר וכמה הגיונית הדרך שבה המוצר מממש יכולת מסוימת. גם כאן הידע שלנו עומד לנו לרועץ.

סיפור ששמעתי ממנהלת מוצר: "ניהלתי פרויקט לפיתוח מכשיר קשר לצבא האמריקאי. אחרי כמה חודשים בפרויקט פתאום נחתה עלי ההכרה שאני עובדת על מוצר שבחיים לא ראיתי. הלכתי למעבדה וביקשתי מאחד המהנדסים להדגים לי את המכשיר. הוא הוריד מכשיר מהמדף, הדליק אותו, כיבה אותו, הדליק שוב, והחל להדגים לי את המוצר. כששאלתי מה זה היה הקטע עם ההדלקה הכפולה, הוא הסביר שככה זה כבר כמה חודשים. אף אחד במעבדה לא חשב שצריך לדווח על זה כבאג.".

למעשה, יש מקרים בהם אנחנו ממש גאים שאנחנו יודעים איך לגרום למוצר לעבוד כשאנחנו (מטאפורית) עומדים על רגל אחת עם יד קשורה לאחור. אנחנו שוכחים שהמשתמשים לא יודעים את כל הטריקים האלה, ואצלם המוצר פשוט לא יעבוד.

מקרה כזה הוא פיתוח הריץ'-רץ', כפי שמתואר בספר "חפצים שימושיים" (The Evolution of Useful Things) של הנרי פטרוסקי: "...וכמו ממציאים קצרי רוח רבים, המיטיבים להכיר את דרכיו של יציר רוחם הבשל, הצליח [הממציא] לגרום שיפעלו באופן מניח את הדעת במעבדה. אבל הלקוחות לא נהגו עדינות רבה כל כך בבן טיפוחיו של הממציא, והשתמשו בו כדרך שהורו להם מודעות הפרסומת [דבר שגרם להופעת בעיות שהמהנדס] לא הצליח לעמוד עליהם, או התעלם מהם בהתלהבותו.".

מבחינה פסיכולוגית קשה לנו להתנתק מהידע הרחב שלנו ולהסתכל על המוצר כמו משתמש תמים. זה עוד אחד מהאתגרים שלנו כבודקים.

"אנחנו גאים שאנחנו יודעים איך לגרום למוצר לעבוד כשאנחנו עומדים על רגל אחת עם יד קשורה לאחור"

דיסוננס קוגניטיבי והטיית האישוש

דיסוננס קוגניטיבי הוא מושג בפסיכולוגיה חברתית ובחקר העמדות, המתאר מצב שבו חשיבתו של האדם מתמודדת עם סתירה וקונפליקט. התיאוריה טוענת כי אצל בני האדם טמון רצון חבוי שמטרתו לשמור על העקביות בין עמדותיהם ותפיסותיהם לבין התנהגותם בפועל. *הטיית האישוש* (Confirmation bias) היא הנטייה לחפש, לפרש, להעדיף, ולזכור מידע המאשש אמונות או השערות קודמות, תוך התעלמות ממידע סותר או כזה שתומך באפשרויות חלופיות. (הגדרות מויקיפדיה).

כשאנחנו נתקלים בתופעה לא צפויה תוך כדי הרצת בדיקה, ההחלטה שלנו איך לפרש את התוצאה מושפעת משתי התופעות הפסיכולוגיות האלה, שבמידה מסוימת קשורות גם לתופעות שכבר הזכרתי. כשקורה משהו במערכת שלגמרי לא מתאים למודל שיש לנו בראש לגבי איך המערכת עובדת (דיסוננס!), ננסה למצוא לזה הסבר שיתרץ את התופעה. בחלק מהמקרים נאמץ הסבר מומצא למה שאירע, במקום לקבוע שיש כאן באג (אם אנחנו מאמינים שהמערכת "בסדר" ועובדת, ההסבר יאפשר לנו לשמור על עקביות בהנחה שלנו לגבי המערכת).

ההשפעה של הטיית האישוש היא שלפעמים נתעלם מעובדות שלא מסתדרות עם התפיסות שלנו לגבי המערכת הנבדקת. לא מתוך כוונה רעה. פשוט לא נשים לב, או נחשוב שדמיינו את זה ובעצם זה לא בדיוק מה שקרה.

תופעות אלה, וגם הקודמות שהזכרתי, גורמות לנו: להתעלם מבאגים, לתת להם הסברים לא נכונים, להמציא באגים שלא קרו, ולעיתים לא להתאמץ יותר מידי למצוא באגים, בעיקר כשמדובר בקוד שכתבנו בעצמנו! (מישהו אמר "קוד של אוטומציית בדיקות" ולא קיבל?).

איך מתמודדים?

הכי קל זה להתעלם מהבעיה. אבל כמו שהפסיכולוגים יגידו לנו, התעלמות לא תשפר את המצב... אז מה כן אפשר לעשות? בתור דבר ראשון, להשתמש במה שעובד לטובתנו:

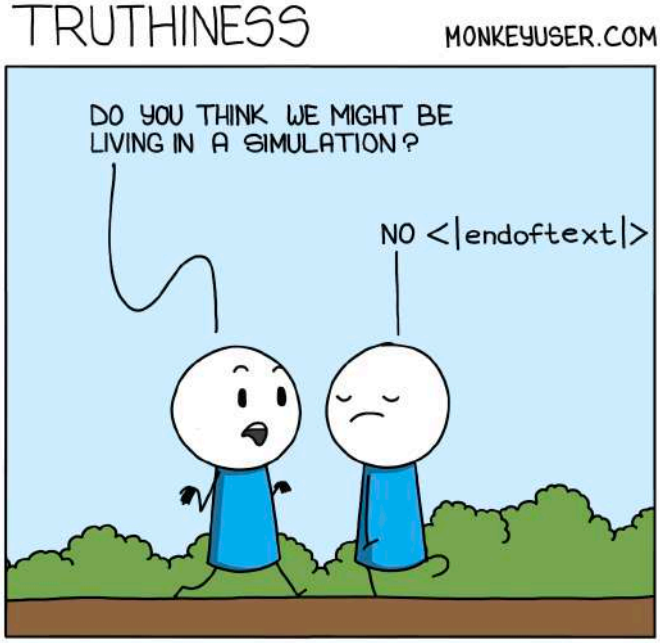
- להגדיר את מטרת הבדיקות "למצוא תקלות" ולא "לוודא שהמערכת עובדת".
- לא להסתפק בבדיקה עצמית לקוד שכתבנו, אלא לתת למישהו אחר לבדוק אותו

"לפעמים נתעלם מעובדות שלא מסתדרות עם התפיסות שלנו לגבי המערכת הנבדקת"

מול התופעות האחרות אפשר:

- להגדיר את סט הבדיקות הראשוני תוך התבססות על הדרישות וציפיות הלקוח, לפני שאנחנו מקבלים הדרכת עומק מהמפתחים או הארכיטקטים על "איך המערכת עובדת". כך נתמודד עם העובדה שהבנה לעומק של הארכיטקטורה חשובה – היא תייצר עוד דברים שנרצה לבדוק – אבל גם תשפיע על מה נחשוב שסביר ומה לא סביר לבדוק.
- שימוש באוטומציה, שלא מושפעת מהחולשות האנושיות - אם כי עדיין ההחלטה (והשגיאות) לגבי מה לבדוק נשארת אצלנו. צריך להיזהר גם בשלב ניתוח התוצאות – שם הסכנה של התעלמות, מציאת הסברים וכו' עדיין קיימת.

מעבר לזה, אני מוצא שהכרת התופעות הפסיכולוגיות מעלה את היכולת שלנו להיות פחות מושפעים מהן, או לפחות לשאול את עצמנו מידי פעם אם ההסבר שמצאנו להתנהגות של המוצר הוא מאולץ ואנו (שוב) קורבן לפסיכולוגיה.



נגיסה מ-TestIL
מפגשים לבודקי תוכנה

עדכונים מערוץ הפודקסט הרישמי של TestIL Podcast מבית ITCB®

את הערוץ מנהלים נתנאל הרוש וקובי יונסי חברים בקבוצת המייעצים (AB) של ITCB®

מוזמנים להאזין לכל הפרקים שיצאו ברבעון האחרון להנאתכם

ניצן גולדנברג

מזה 8 שנים בתחום בדיקות התוכנה, תפקיד נוכחי מהנדס בדיקות בכיר בחברת SeatGeek, מנהל קבוצת ה-AB של ארגון ITCB® המוביל הראשי של קבוצת המיטאפ TestIL, מרצה בקורסים לבודקי תוכנה

פרק #21 – איך עושים בדיקות על משחקים?
בפודקאסט שלפניכם אביתר מספר טיפה מן הים הגדול של התהליך אותו עבר בשנים האחרונות, אתגרים בעולם הגיימינג, הצלחות, חשיבה מחוץ לקופסא ובעיקר הצורך הגדול בלהישאר מעודכן בטכנולוגיות מסביב לעולם.

פרק #22 – בואו להכיר את Jam.dev
דני ואירטפה מספרים למה הם הקימו את החברה, איך הגיעו לשם Jam ומה צופה העתיד של החברה. ניצן ונתנאל מספרים להם על החוויה אישית שלהם בשימוש יום יומי במוצר למעלה משנה, למה הם כמשתמשים מצפים לראות בהמשך (סוג קטן: דיווח באגים במובייל בקליק אחד) ואיך היום הם רואים את עצמם מדווחים באגים ב-Web ללא הכלי הזה.

פרק #23 – בדיקות חוקרות עם עמית ורטהיימר
עמית ורטהיימר, ירושלמי, בודק תוכנה מזה כ-12 שנה, חבר ב-Association for Software testing, עורך (כשמרשים לו) במגזין "עולם הבדיקות", מתחזק (בקושי) בלוג על בדיקות תוכנה, מדבר מדי פעם בכנסים, ומחזיק בדעה נחרצת על כל דבר. כיום עובד ב-Deep Instinct..
בפרק הזה עמית מדבר על בדיקות חוקרות, מתי אנחנו משתמשים בהם ומתי צריך אותם.
עמית מספק לנו מספר מקורות לא קטן של מידע אודות בדיקות החוקרות.

פרק #24 – סטארטאפ VS קורפורייט עם ג'ני רויטמן
בפודקאסט שלפניכם נתנאל וג'ני דנים בהבדלים בין סטארטפ לקורפורייט, הן בהיבטים של בדיקות ובכלל. על תרבות אירגונית ותהליכי קבלת החלטות, על כלים וטכנולוגיות והיקף הפרויקטים. במה סטארטאפים יכולים ללמוד מקורפורייט ולהפך והיכן כדאי להתחיל לעבוד וללמוד אם זו העבודה הראשונה שלנו.

פרק #25 – שורטקאסט "טכניקת בדיקות מבוססת ניסיון – קובי יונסי
אחת השיטות המוכרות בטכניקות בדיקה מבוססות ניסיון היא שיטת ניחוש שגיאות.
שיטת זאת מבססת את הניחוש על היכרות מוקדמת של הבודקים עם המערכת או עם מערכות מחשוב דומות.
היכרות שיכולה לאפשר להם לנחש מראש איפה יכולות להסתתר תקלות.
בשורטקאסט קובי יונסי יתאר את הטכניקה ואופן יישום השיטה הלכה למעשה.

פרק #26 – בדיקות ריגרסיה עם ניסים סער אריאל
ניצן גולדנברג ונתנאל הרוש עושים ראיון משותף עם ניסים סער אריאל, Team Lead בחברת X-trodes.
בפודקאסט הזה ניסים מדבר על אחד הדברים החשובים בניהול בדיקות תוכנה לפני "שעולים לפרודקשיין" - בדיקות רגרסיה (נסיגה).

פרק #27 – עושים QA לקריירה – מנהל או מנהיג עם איילת מלמד כהן
פעם דיברנו על מהו הדבר העיקרי שמבחין בין מנהל למנהיג ואיך אנחנו יכולים לבטא את המנהיגות שקיימת בנו וליצור השפעה.
דיברנו על הצורך של כולנו להשפיע בעבודה ומחוצה לה - על עצמינו, על איך אנחנו נתפסים, על החלטות שמתקבלות ועל כיוון פיתוח הקריירה שלנו. כמה הדרך לעשות זאת חוצה את גבולות הסמכות הפורמלית או הטייטל, ועוברת דרך מנהיגות עצמית.

אם גם אתם מעוניינים להשתתף בפודקסטים, אנא צרו עימנו קשר במייל: קישור לערוץ הפודקסט שלנו

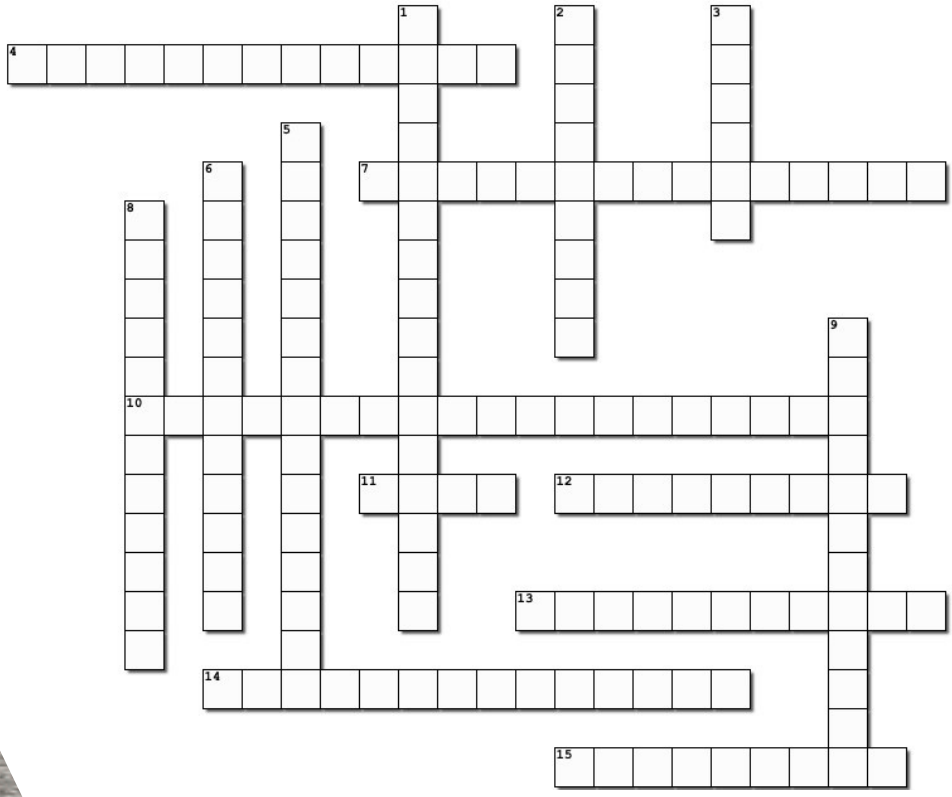
בודקים FUN בכיף

Test Automation Engineer

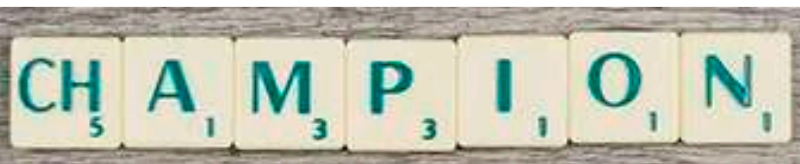
התבלבלו לי המילים

1. aioicpnlapt gmarnpoirgm caetinrfe _____
2. tzooahanturii _____
3. sisanticlfcaio rete _____
4. ngodci aarstdtn _____
5. toaoinaumt deco eectfd sndyeit _____
6. lci nstgeit _____
7. earcogve _____
8. adat rinedv esigntt _____
9. engggiubd _____
10. vrderi _____
11. utafl tiecionjn _____
12. caplahirg ruse eienfcrat _____
13. arilne iritcnspg _____
14. emldo vaegerco _____
15. rpee weirev _____
16. resduutctr sitngte _____
17. ustb _____
18. etts utnooamtia _____
19. estt srneash _____
20. tset kooh _____

תשבץ אוטומציה



שם הזוכה יפורסם בגליון מס 38
יש לשלוח את הפתרון למייל
MAGAZINE@TESTINGWORLD.CO.IL



Across

4. Permission given to a user or process to access resources
7. process of intentionally adding a defect to a component or system
10. tree showing equivalence partitions hierarchically ordered
11. skeletal or special-purpose implementation of a software component, used to develop or test a component
12. customized software interface that enables automated testing of a test object
13. review of a software work product by colleagues of the producer of the product for the purpose of identifying defects
14. The coverage of model elements
15. process of finding, analyzing and removing the causes of failures in software

Down

1. A simple scripting technique without any control structure in the test scripts
2. Data created or selected to satisfy the execution preconditions and inputs to execute one or more test cases
3. software component or test tool that replaces a component
5. standard that describes the characteristics of a design or a design description
6. test environment comprised of stubs and drivers needed to execute a test
8. A risk directly related to the test object
9. The required state of a test item and its environment prior to test case execution

עורכי הגיליון

תמרה מוסונובה

בודקת תוכנה ואינטגרציה בחברת Varonis. בעלת תואר בתקשורת וקולנוע והסמכות פיתוח אפליקציותWeb של מיקרוסופט, כך משלבת חשיבה יצירתית ואנליסטית בעבודה. בונה אתרים ועורכת סרטים כתחביב. שחקנית כדורשת בליגה ארצית, תופסת כדורים ובאגים מקצועית.

אפרת וינברג

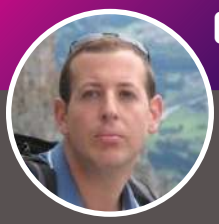
עוסקת בבדיקות תוכנה קרוב ל-20 שנה. עבדה במספר ארגונים בתפקידי בדיקות וניהול בדיקות. בשנים האחרונות עוסקת בפיתוח והוראת קורסים בבדיקות תוכנה ונושאים נוספים

אסתר צבר

מהנדסת (M.Sc.) בעלת 23 שנות ניסיון בפיתוח ובדיקות תוכנה, מתוכן 11 שנים בניהול QA בחברות ECI ו BMC - ובנוסף חברה ב-Advisory Board של ®ITCB הארגון הישראלי להסמכת בודקי תוכנה. בתשע השנים האחרונות – יזמית ומנהלת של AQA המכשירה ומשלבת אנשים עם תסמונת אספרגר בעבודה בהייטק כבודקי תוכנה.

משה מאמיה

בעל 17 שנות ניסיון כמהנדס, מתוכן מעל עשר שנות ניסיון ניהולי, מתמחה בפיתוח אוטומציה ובבדיקות ביצועים. עובד מעל 5 שנים ב-HP כמנהל קבוצות QA ו-DevOps. חבר מייעץ למועצת מנהלים של ®ISTQB ומרצה בפקולטה להנדסת תוכנה במכללת SCE.

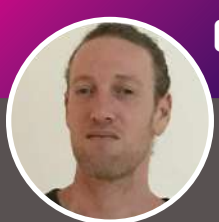


עמית ורטהיימר

בודק תוכנה ב-Deep Instinct.

רחל ברוך

עובדת כבודקת תוכנה בכלל ביטוח. לאחר הפסקה של עשור מעולם התוכנה חזרה למקצוע הכי אהוב אליה. כשעובדים בעבודה שנהנים בה הזמן טס.

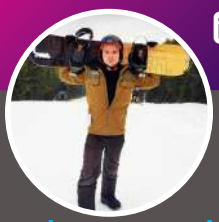


דור רוזנברג

לפני כמה שנים החלטתי לעשות שינוי במקצוע שלי. לאחר מספר תחומי לימוד הכי הרבה התלהבתי מלימודי בדיקות תוכנה. כרגע עובד בשרות לקוחות בישראכרט.

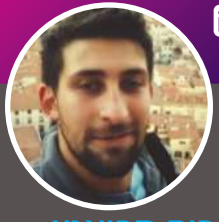
שירה נוסבוים

הייטקיסטית ואמא במשרה מלאה, בדקות הבודדות שנשארות ביום בלוגרית אפייה.
בעלת תואר ראשון במדעי המחשב ובכימיה, מעל 10 שנות ניסיון כמפתחת תשתיות אוטומציה וכלים אוטומטיים בחברות גדולות, בסטארטאפים שונים בתעשייה במגוון תחומים. מובילת תחום, מרצה ומפתחת קורסי אוטומציה. בשבילה החיים זה לא מספיק.



אלכסנדר זבולוקו

מפתח תשתיות אוטומציה בחברת SeatGeek. מתמחה ב-DevOps וטכנולוגיית ענן לעומק. מנצל את הידע והמיומנות לפיתרון בעיות ולהביא לחדירה מירבית של טכנולוגיות חדשות בסביבת התשתיות.
מחבר את המקצועיות עם התשוקה לטייל ברחבי העולם, מתרגש לחוות חוויות ותרבויות חדשות בזמן עבודה על מייזמים פרטיים.

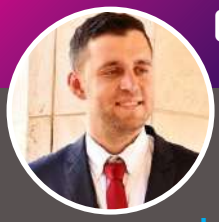


רוביק סביאנץ

בודק תוכנה, נמצא בתחום מעל 4 שנים את דרכו התחיל בחברת CARAMBOLA נכון להיום מחזיק את מערך הבדיקות בחברת OOLO בזמן הפנוי - ספורט, טיולים ומחשבים

שי ביטון

על 12 שנות ניסיון בפיתוח אוטומציות ובדיקות.
עובד כיום ב- Qwilt.



אלכס קומנוב

בעל מספר שנים בתחום הבדיקות האוטומטיות. עובד כמפתח בדיקות ותשתיות אוטומציה בכיר בחברת SeatGeek נשוי+1 חובב נגינה על גיטרה וטיולים.

טל פאר

בעל ניסיון של יותר מ-25 שנים כבודק ומנהל בדיקות במגוון חברות וטכנולוגיות. כיום טל הוא יועץ בכיר ב-Grove Software Testing, חברת הדרכה מובילה בבריטניה וספקית של חומרי הדרכה.
טל פעיל ב-®ISTQB והיה חבר בצוות המנהל של הארגון במשך 6 שנים. טל חבר בצוות המנהל של ®ITCB.

מגזין עולם הבדיקות