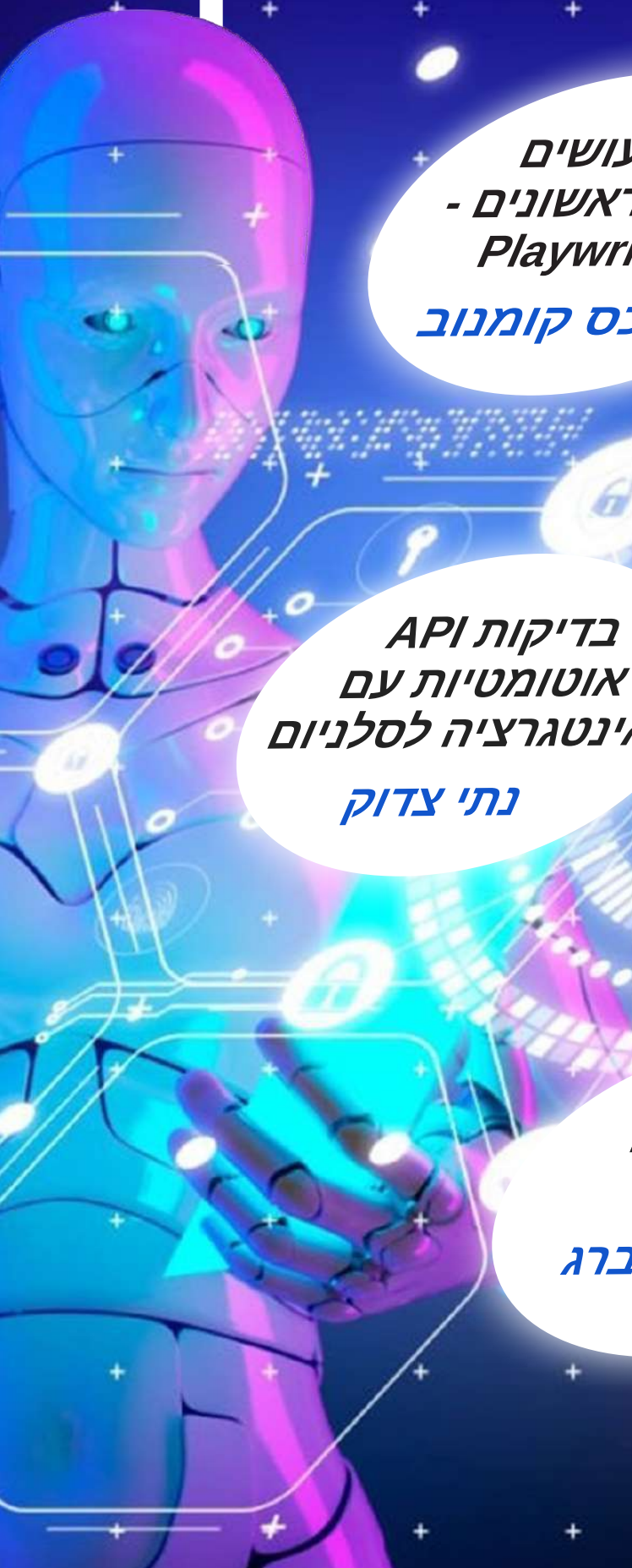
מגזין
עולם הבדיקות
רבעון ראשון 2023 | גיליון מס' 32
www.testingworld.co.il



איך עושים צעדים ראשונים - Playwright
אלכס קומנוב

ראיון עם מנהלת בדיקות - מאיה צמח
ניצן גולדנברג

בדיקות API אוטומטיות עם אינטגרציה לסלניום
נתי צדוק

ChatGPT
יוני פלנר

מה הסקרים אומרים
ניצן גולדנברג

דבר העורך
ניצן גולדנברג

ניצן גולדנברג

מזה 7 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,

"I knew that if I failed I wouldn't regret that, but I knew the one thing I might regret is not trying."

Jeff Bezos

פנתה אלי בפורום של **TestIL** מישהי שביקשה ממני ייעוץ לגבי משרה שהיא ראתה, המשרה מיועדת לבוגרי קורס בדיקות ללא ניסיון מעשי בבדיקות אבל ידע טוב בשאילתות SQL. היא ביקשה להתייעץ איתי ושאלה האם אני חושב שהיא יכולה להציע את עצמה למשרה למרות שהיא לא חזקה ב-SQL.

שאלתי אותה חזרה האם יש לה קושי לחזור על החומר וללמוד לפני הראיון? תשובתה הייתה שאין שום בעיה ולכן, עניתי לה שמכל ראיון עבודה ניתן ללמוד ולהשתפר, גם אם לא צולחים כל ראיון. הרי אם לא ננסה, ככל הנראה נצטער שלא ניסינו כלל.

לכם קוראים וקוראות יקרים, אני אומר שאם תכשלו במשהו שנסיתם, באם זה זה ראיון שלא צלח או בדיקה שתכננתם ולא צלחה או סתם משחק ששיחקתם ולא ניצחתם, אולי תתאכזבו מעט וזה בסדר לא להצליח תמיד, אבל אני יכול לומר בבטחון מלא שהייתם מתאכזבים יותר אם לא הייתם מנסים כי כל עוד אתם יודעים שנתתם מעצמכם הכל בניסיון אז אתם המנצחים. כמו שאומרים, ה"לא" נמצא שם תמיד, בואו ננסה להשיג את ה"כן".

מונח לפניכם גליון מס 32 של מגזין "עולם הבדיקות".

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

אלכסנדר קומנוב ממשיך להביא לנו את סודות Playwright במאמר מעניין "Playwright - איך עושים צעדים ראשונים"

כתבה מרתקת בנושא בדיקות API - "בדיקות API אוטומטיות עם אינטגרציה לסלניום"

יוני פלנר מנצל את המומנטום הלוהט בשוק כיום עם מאמר "מי עדיין לא שמע על ChatGPT"

הטור "מה הסקרים אומרים" חוזר אלינו לאחר תקופה ארוכה של 3 שנים

אנו נשמח לקבל בקשות לנושאים חדשים ומעניינים למאמרים, צרו עימנו קשר במייל: **Info.testingworld@gmail.com**

קריאה מהנה,
ניצן גולדנברג

הודעות

- האתר החדש של ארגון ITCB עלה לאוויר ואתם מוזמנים להכנס לכתובת **www.ITCB.org.il** ולהתרשם

מו"ל
Israeli Testing Certification Board
ITCB®

ניהול המגזין
יאן ברון, *iMDsoft*

ניהול התוכן
קובי הלפרין, *Nokia*

עורך ראשי
ניצן גולדנברג, SeatGeek

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

יצירת קשר
אימייל:
info.testingworld@gmail.com

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

ראיון עם מנהלת בדיקות
מאיה צמח, Change Healthcare

מאיה צמח

נשואה לצחי ואמא לשלושה ילדים מקסימים: עילאי, מיקה ויואב. עוסקת בעולם האיכות והבדיקות למעלה מ-14 שנה. לתפקידי היום - QUALITY MANAGER בחברת Change Healthcare - צ'יינג' הלת'קר, הגעתי לאחר מספר שנים בתעשייה כבודקת וכמנהלת צוות. בחברה התחלתי כבודקת בקבוצת הפיתוחים הטקטיים ללקוחות, לאחר מכן עברתי לניהול צוות הבדיקות ומשם לתפקידי הנוכחי. התפקיד כולל בין היתר:

- שיפור תהליכים בתחום הפיתוח והבדיקות, פעילויות וניהול איכות הגרסאות
- אישור גרסת מוצר הקרדיולוגיה לכל סוגיה
- מנטורינג לקבוצת הבודקים
- ניהול קבוצת בודקים ייחודית המספקת שירותים לקבוצות פנימיות DBA/DevOps/SysEng
- סנכרון ותיאום האינטגרציות והשותפויות של המוצר
- קידום יוזמות וחידושים הקשורים לשיפורי איכות
- קביעת מדדי איכות וניטור ביצועם על ידי כלים מתאימים

במה עוסקת הקבוצה וכיצד היא בנויה?

Change Healthcare הנה חברה הפועלת בלב העשייה של מערכת הבריאות האמריקאית.

החברה עובדת עם רוב בתי החולים והמבטחים בארה"ב.

העשייה טומנת בחובה משמעות וערך (עיסוק בתחום הרפואי, אפליקציות רפואיות, פתרונות בנקודות הקריטיות לחולה – עולם ההדמייה, קריולוגיה).

המרכז בישראל עוסק היום בפיתוח מוצרי תכנה וניטור לשוק ההדמיה (קרדיולוגיה ורדיולוגיה) המעניקים פתרון מקיף לניהול מידע והתממשקות למערכות שונות בבתי חולים. החברה מפתחת מערכות CPACS לניהול כל תהליך העבודה של קרדיולוגים בדיאגנוזה של תמונות קרדיולוגיות (אקו, צינתורים MRI CT), מערכת הימודינמיקה לניטור חולים בזמן צינתורים ומערכת Management ECG לפענוח פרוצדורות אק"ג.

בשנת 2018 הקמנו מרכז חדשנות בישראל המשרת את כל Change Healthcare העולמית שמטרתו פיתוח מוצרים חדשנים בשת"פ עם יחידות עסקיות אחרות של החברה.

החברה מובילה ופועלת בעולם התוכן הרפואי, וככזו משמשת "בית-ספר" לעובדים בה בכל התחומים בהם ניהול מוצר, פיתוח, רגולציה, תמיכה וכו.

קבוצת הבדיקות כוללת מהנדסים אשר מתמחים בבדיקות פונקציונאליות, אוטומציה, ביצועים ותשתיות.

מהם האתגרים הייחודיים של קבוצת הבדיקות שלכם וכיצד אתם מתמודדים איתם?

מוצרינו הנו מוצר רפואי וככזה אנו עומדים בפני לא מעט אתגרים. הדרישות העולות בתחום זה, רמת התיעוד הנדרשת, צרכים רגולטוריים, רמת האבטחה הנדרשת והצורך באיכות מוצר גבוהה.

אתגרים אלו מביאים אותנו לחשוב ולפעול יום יום על מנת לשמר ולעמוד בפניהם, ברמת תהליכים הפיתוח ושחרור הגרסה.

ללא ספק כל אדם, בפרט אם הוא חולה, היה רוצה שהמערכות שמשפיעות על הטיפול בו יהיו איכותיות, מאובטחות ובטוחות.

תהליכי הבדיקות כוללים שקיפות ושיתוף רוחבי של כל הגורמים הרלוונטיים בארגון על מנת לייצר ולספק תוכנית בדיקות מקיפה ומימושה בצורה הטובה ביותר.

תוכנית הבדיקות נבנית תוך מתן דגש על סביבות ייחודיות הקרובות לסביבת לקוח (בתי חולים), מגוון רחב של בדיקות מקיפות הכוללות בדיקות ידניות ואוטומטיות, ראייה מערכתית, דרישות אבטחה, דרישות רגולטוריות ותיקוני באגים.

כיצד את מניעה (Motivate) את הבודקים ומה עוד היית רוצה לעשות?

אני מניעה את העובדים על ידי יצירת קהילה פנימית של הבודקים בארגון. קהילה שמטרתה שיתוף ידע וניסיון בתחום הבדיקות ומענה לצרכי הפיתוח והאתגרים הקיימים בפיתוח ובבדיקות תוכנה רפואית.

הקהילה מתרכזת בנושאים הבאים:

- שיתוף הידע הקיים בפלטפורמות תקשורת השונות המייצר חיבור של העובדים בין קבוצות בדיקות שונות למען מטרה משותפת, העלאת המוטיבציה האישית וקידום איכות הבדיקות.
- קיום ישיבות חודשיות בהם דנים בנושאים שונים כגון סוגי בדיקות, אתגרים ושיפור תהליכים.
- הדרכות חיצוניות ופנימיות.
- עמידה במדדי איכות וניטורם במערכת.

הייתי רוצה לשפר את נושא האוטומציה הן מבחינת תשתיות והן מבחינת פוקוס והשקעה.

כיצד הנך פועלת להעשרת הידע של הבודקים? (הן מבחינה מתודולוגית והן מבחינה טכנית)?

העשרת הידע נחלקת למספר תחומים:

- מנטורינג של מנהלי הקבוצות הכולל מידה, הטמעה וחניכה בעולמות הניהול על-פי צרכי הארגון.
- מנטורינג על ידי מנהל ישיר
- קורסים והרצאות בתחום הניהול

מה היית ממליצה לבודק תוכנה שנמצא בתחילת הקריירה שלו?

אני ממליצה על מספר טיפים חשובים לקידום הקריירה:

- בניית ערוצי תקשורת בריאים בכתב ובעל פה עם המנהל הישיר, עם המפתחים,

ראיון עם מנהלת בדיקות מאיה צמח, Change Healthcare

מנהלי המוצר ועם כל ממשק מולו עובדים באופן ישיר או עקיף היכולת לתקשר, להרים דגל במידת הצורך, להסביר בעיה או לסכם תוכנית בדיקות היא חשובה מאין כמוה ותורמת לשיתוף פעולה פורה, קידום המוצר ואיכותו ועמידה בלוחות זמני הפרויקט

- שקיפות ויסודיות אשר משמעותם לספק מידע רב, מהימן ומקיף אודות הבעיה, הסטטוס, דרך חקירה, הפערים ופתרונות אפשריים
- פיתוח כישורים טכנולוגיים כגון: לימוד כלים, טכנולוגיות או שפות תכנות
- השתתפות בהרצאות, קורסים ומיטאפים בעולם התוכן הרלוונטי
- הצטרפות לקהילת בדיקות בארגון ומחוצה לו
- פיתוח מיומנויות אישיות כגון: אסרטיביות, יכולת ביטוי, עבודת צוות ולקיחת אחריות

אנא שתפי בכמה מההישגים העיקריים של קבוצת הבדיקות

קבוצת הבדיקות על כל חלקיה הביאה תוצאות מדהימות במהלך השנים האחרונות.

תהליכי הבדיקות וההשקעה בתשתיות ובהדרכות וכמובן פוקוס מתמיד על איכות הביאו את בדיקות המוצרים שפותחו בגרסאות האחרונות לאיכות גבוהה.

שיפור קבוע בכיסוי בדיקות, ולידציה על סביבות לקוח שונות ומתן דגש על פתרון באגים בכל הרמות מוכיחים את עצמם.

כמות הבאגים הנמוכה שנמצאה בשטח לאחר הטמעת המוצר מעידה בין היתר על רמת האיכות.

בנוסף תהליך ההתקנה השתפר משמעותית והלקוחות מפרגנים על כמות הזמן שהצטמצמה ועל היעילות של המערכת.

מה הן התובנות שלך לגבי עולם הבדיקות?

עולם הבדיקות הוא עולם דינאמי ומשתנה.

היום בודק נדרש להיות מעורב מתחילת התכנון, לתעדף בדיקות לפי חשיבות, לבצע רגרסיה מתאימה במידת הצורך,

להיות גמיש בהתאם לשינוי תוכן שמתרחשים במהלך האיטרציה, ולעיתים להתחיל לפתח בדיקות ללא דרישות מלאות.

למרות הכלים האוטומטיים וכלי ה-AI, בודקים ידניים ואוטומטיים ימשיכו להחזיק את חוד החנית של איכות המוצרים והם מחויבים למקצועיות, יסודיות ומעורבות גדולה ברמת הכרת המוצר והשפעתו על הלקוח ועל תהליכי הבדיקות.

הגישה כלפי אנשי בדיקות השתנתה והם כיום נתפסים כאנשי איכות שמעורבותם.

באלו פעילויות קהילתיות את ועובדייך השתתפתם לאחרונה ומהן התובנות שלקחתן מהן?

בימים אלו אני עובדת על הרצאה במסגרת קורס בדיקות לחברת AQA.

AQA היא תכנית חדשנית להכשרת ושילוב אנשים מהספקטרום האוטיסטי בתפקודים הגבוהים, בבדיקות תוכנה בהייטק. התכנית החלה בפעילותה בשנת 2008 ומאז שילבה בהצלחה יותר מ-160 מבוגריה ב-98 חברות הייטק שונות.

הפרויקט מכוון לאנשים על הרצף האוטיסטי שהם בתפקוד גבוה למעט מגבלות מסוימות בתחום החברתי. בנוסף לכך, יש להם יתרונות ייחודיים:

- יכולת חזותית יוצאת דופן, זיכרון צילומי, ותשומת לב לפרטים
- פחות מוסחים ע"י צרכים חברתיים, בעלי יכולת ריכוז ומיקוד ולכן הספק העבודה שלהם גבוה
- חיבור חזק למערכות שיש בהן לוגיקה וחוקים ברורים
- מעדיפים עבודות חוזרות ונשנות
- חשיבה ייחודית וידענות רבה

האתגר החברתי הוא למצוא את הדרך לשלב אנשים אלו במעגל העבודה במקצוע נדרש ולתת ביטוי ליכולותיהם.
יתרונות אלו נותנים ערך מוסף משמעותי בביצוע משימות של:

- בדיקות רגרסיה
- בדיקות נתונים
- משימות שמחייבות דייקנות, קפדנות, תשומת לב לפרטים, מונוטוניות

בתחילת 2022 התחלנו שת"פ עם ארגון AQA במטרה לקלוט בוגרים לטובת בדיקות ידנית לפרוייקט Cloud PACS

נכון להיום אנחנו עם 3 בוגרים שכותבים מתחזקים ומריצים VnV בכל גרסה.

פספורט קבוצתי

Change Healthcare

סוג המוצר הנבדק: וב משולב דסקטופ (תוכנה וחומרה)

גודל קבוצת הבדיקות: כ-20 מהנדסי בדיקות

וותק הבודקים: הוותק נע בין שנה ל-12 שנה

סוגי בדיקות: בדיקות עשן (Smoke), שפיות (Sanity), ריגרסיה, שימושיות, אבטחה, ביצועים, התקנה, שליליות, ולידציה קלינית ובדיקות אוטומטיות (Unit/Integration/Component)

שיטת עבודה: קנבן ואג'יל סקראם

כמות המוצרים הנבדקים וקצב שחרור גרסאות:
שחרור גרסה למוצר הקרדיולוגיה מתבצע בין רבעון לחצי שנה

המראיין: ניצן גולדנברג
מזה 7 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

אפקט הגשושית

כמי שמשקיע לא מעט בלימוד תיאוריה של טכניקות בדיקה שונות, אני נאלץ להודות שהרבה פעמים הבאגים שאנחנו מוצאים לא ממש מתיישבים עם התיאוריה. אומנם קורה מידי פעם שמוצאים באג שנובע מטעות של off by one ומכשיל בדיקה של מקרה קצה, אבל זה קורה הרבה פחות ממה שהיית מצפה לאור כמות האזכורים של טכניקת ניתוח ערכי גבול (boundary value analysis) בספרים. לכן אני תמיד נהנה כשמוצאים באג ש"מתיישב" יפה על התיאוריה.

לאחרונה חוויתי במסגרת עבודתי מקרה שהוא סוג של אפקט גשושית (probe effect) – שגם היא תופעה שיותר מדובר עליה מאשר נתקלים בה בצורה ברורה ביום יום. אז אתחיל בסיפור המעשה, אחר כך קצת תיאוריה, ואז עוד דוגמאות מהעבודה – שגם הם לא מתיישבות על התיאוריה במאה אחוז.

הבאג הנעלם

בימים אלה אני עובד על מערכת שאחד מהחלקים שלה הוא מכשיר קטן שמופעל מרחוק. על המכשיר יש מתג הפעלה. אחרי התקנת הגרסה האחרונה, המכשיר לא נדלק. כלומר, נורת ה- LED שמסמנת שהמכשיר "הבין" שהפעילו אותו אכן נדלקה, אבל זהו. המכשיר נשאר תקוע שם, ולא המשיך לשלבי האתחול הבאים.

שמחה גדולה! מצאנו באג! עכשיו רק צריך להוציא קצת לוגים ואפשר לסמן חריץ נוסף ברשימה של critical bugs.

על מנת לקבל לוגים, יש בגרסת הפיתוח של המכשיר יציאה טורית, שאליה ניתן לחבר טרמינל פשוט ולקבל חיווי של תהליך האתחול של המכשיר. חיברתי הכל כיאות, הדלקתי את המכשיר, ו... המכשיר נדלק, סיים את האתחול כמו ילד גדול, והיה מוכן לשימוש.

לך עכשיו תוכיח שיש לך אחות.

מסתבר שגם חיבור המטען ליציאה הטורית מאפשר למכשיר להדלק כראוי. עדיין לא ברור בדיוק מה העניין, אבל הניחוש שלי הוא שיש משהו גבולי במעגל האלקטרוני, וחיבור עומס על היציאה הטורית מסדר את זה איכשהו. וכמו בבדיחה על כמה אנשי תוכנה צריך להחליף נורה, הבעיה הפכה לבעיית HW.

מה שכן, זו דוגמה (לא קלאסית!) לאפקט הגשושית.

אפקט הגשושית (Probe effect)

על פי **ויקיפדיה**, אפקט הגשושית הוא "שינוי לא מכוון בהתנהגות של מערכת, הנגרם כתוצאה ממדידת התנהגות המערכת"

יש כמה דוגמאות קלאסיות:

א) תוספת של debug prints לקוד, לצורך זמני (למשל, לעזור בתהליך debugging) או באופן קבוע על מנת לאפשר מצב הפעלה מיוחד לאיתור תקלות לאחר שחרור המוצר ללקוחות. תוספת זמן העיבוד שההדפסות מכניסות משנות את התנהגות המוצר, ותופעות שנצפו ללא ההדפסות, נעלמות כשההדפסות מופעלות (או להיפך: ההדפסות גורמות לתקלות פונקציונאליות במוצר).

ב) מכשור (instrumentation) של קוד המקור, על מנת למדוד כיסוי קוד (code coverage). בלי להיכנס לפרטים הטכניים, פעולת המכשור מוסיפה על הקוד המקורי עוד פקודות להרצה. גם כאן יש תוספת זמן עיבוד. למרות שפתרונות מכשור מודרניים מכניסים ממש מינימום של תוספת קוד, זה עדיין תוספת – ואיתה ההשפעה על מהירות הרצת התוכנה.

ג) הרצת כלים במקביל לתוכנה הנבדקת על מנת לאסוף מידע על ביצועי התוכנה. העובדה שעל אותו מחשב מורצות עוד תוכנות בזמן הבדיקה, שחולקות משאבים עם התוכנה הנבדקת, משפיעה על זמן המעבד שמוקצה לתוכנה הנבדקת, על מהירות זמינות הזיכרון ואפילו על טמפרטורת המעבד. לכל אלה יש השפעה – לעיתים מינורית אבל בכל זאת השפעה – על הביצועים הנמדדים.

מורשת קרב (1) – בדיקת אפליקציית ראייה ממוחשבת

הסיפור הבא מתאר עד כמה אפקט הגשושית יכול להתרחק מהמקרה הקלאסי – ועד כמה צריך לשים לב מה משפיע על תוצאות הבדיקה (וזה לא רק הקוד הנבדק!).

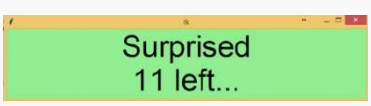
> **"אפקט הגשושית: שינוי לא מכוון בהתנהגות של מערכת, הנגרם כתוצאה ממדידת התנהגות המערכת"**

לפני כעשר שנים התבקשתי לתת חוות דעת מהירה וכללית על ישום לזיהוי רגשות על פי וידיאו של הפנים. הקוד ידע להבחין בארבע רגשות: כעס, שמחה, עצב, הפתעה. על כל תמונה של קלט, עבור כל הבעה, היישום נתן את רמת הבטחון באחוזים שזו ההבעה שנראית בתמונה (כלומר ארבעה מספרים, שכל אחד בין 0% ל-100%. סכום המספרים לא היה צריך להתכנס ל-100%).

על מנת לבדוק את היישום, החלטתי להקליט כמה סרטונים שבהם מישהו יושב מול המצלמה ומדגים את הרגשות האמורים. בגדול – כמו שחשבתם: חיוך רחב עבור שמחה, העוויה של כמעט-בכי עבור עצב, גבות ופה מכווצים לכעס, ופה פעור וגבות מורמות להבעת הפתעה. הנחנתי שהבעות פנים מהירות מידי לא יקלטו טוב על ידי התוכנה, וכיוון שבשלב זה רק רצינו לוודא שהתוכנה עובדת על מקרים חד משמעיים, צריך היה לוודא שכל הנבדקים מציגים למצלמה כל הבעה פחות או יותר אותו זמן. כדי להבטיח את זה כתבתי תסריט קצר בפייתון שהציג לנבדק הוראה איזה הבעה לעשות, והחליף את ההוראה כל חמש שניות. הנבדקים ישבו מול מסך עליו הופיעו ההוראות, ומעל המסך הייתה המצלמה שהקליטה אותם. על מנת שהמרחק מהמצלמה לא ישחק תפקיד בתוצאות הצבתי את הכיסא בו ישבו הנבדקים במרחק קבוע מהמצלמה, והדגשתי לנבדקים שיש לשמור את הראש והגוף צמודים למשענת הגב.

התסריט הציג לנבדק רצף של 12 הבעות, בסדר אקראי (שלוש מכל סוג הבעה) כך שיכולתי גם לבדוק אם התוכנה מושפעת מסדר ההבעות (למשל, האם זיהוי שמחה מצליח יותר פעמים אם ההבעה הקודמת הייתה של עצב). התסריט שמר בקובץ את רצף ההבעות שהציג, על מנת לאפשר השוואה בין מה שהנבדק הדגים לבין מה שהתוכנה זיהתה.

הנה דוגמה למה שהתסריט הציג:



במקרה זה על הנבדק היה להדגים הפתעה - עד שההוראה התחלפה להוראה חדשה.

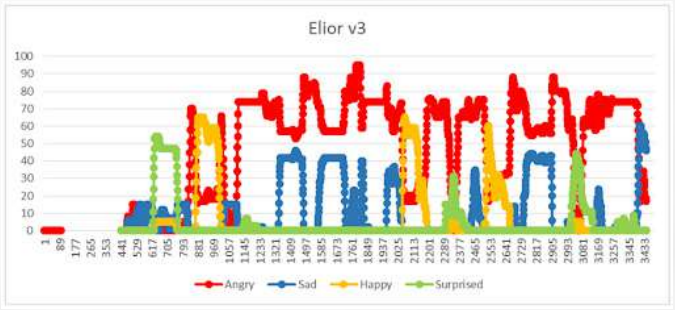
לאחר כל הקלטה, העברתי את הסרטון ליישום ובדקתי אם ההבעה שהיישום זיהה היה נכונה. הנה אחת התוצאות, עבור רצף של שתים

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

עשרה הבעות. ציר ה-X מסמן את המספר הסדרתי של התמונה בווידאו. ציר ה-Y מסמן את רמת הבטחון של התוכנה בהבעה שזוהתה.



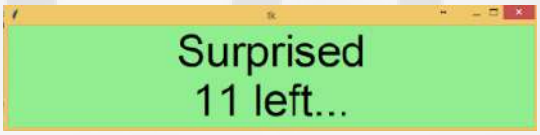
מיד ראיתי שיש בעיה: התוכנה מזהה הרבה יותר מידי פעמים כעס. למעשה, זיהוי כעס קורה עבור כמעט כל הבעה. זה היה לי קצת מוזר. מילא עצב שיזוהה ככעס – זה טיפה קרוב. אבל שמחה, או הפתעה?

"המון דברים יכולים להשפיע על תוצאות הבדיקות"

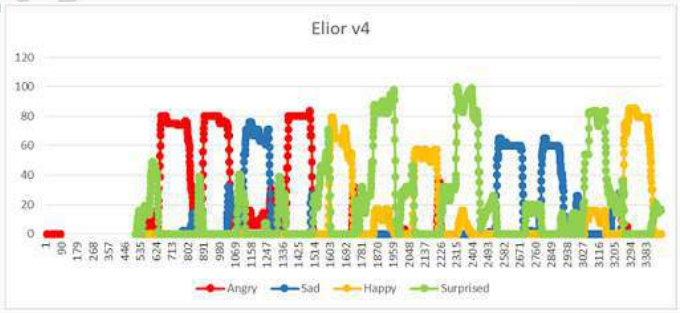
כדי לברר מה קורה, צפיתי בהקלטות הוידאו. אולי הנבדקים לא הבינו מה נדרש מהם והרבו לטעות? הספיקה לי צפייה בשניים שלושה סרטונים להבין מה קרה. מסתבר שההוראות על המסך היו בגודל קטן מידי. מצד אחד הנבדקים התקשו לקרוא את ההוראות מהמרחק שהגדרתי, ומצד שני ביקשתי שישמרו את הראש במרחק נתון מהמסך. אז הם קצת צמצמו את העיניים כדי לראות, וקצת כווצו את הגבות...

כעס!

שינוי קטן בתסריט ייצר הוראות בפונט גדול יותר:



והתוצאות בהתאם – שלוש הבעות מכל סוג, בהופעה אקראית. לא מושלם, אבל גם לא על הפנים...



מורשת קרב (2) – מאיפה הקלט

דוגמה נוספת שבה נתקלתי קשורה גם היא לבדיקות במערכות ראייה ממוחשבת. אוטומציה של בדיקות במערכות אלה מחייבת יכולת הזנת תמונות מקבצים ולא ישירות מהמצלמה.

נניח שאנו בודקים מערכת לזיהוי פנים. לגמרי לא יעיל להעמיד כל סבב בדיקות אנשים מול המצלמה על מנת לוודא שהמערכת זיהתה אותם. במקום זה מקליטים סרטונים של מספר גדול של אנשים, ומזינים אותם למערכת כאילו הם מגיעים מהמצלמה.

"שיטת הבדיקה משפיעה ישירות על הערכים הנמדדים"

ה"כאילו" הזה טומן בחובו הזדמנויות רבות לאפקט הגשושית: האם התמונות מוזנות לתוכנה באותו הקצב שהם מגיעות ממצלמה? האם התמונות עוברות אם אותו קדם-עיבוד (pre-processing) שמבוצע על תמונות מהמצלמה? אם לצורך ביצוע קדם-עיבוד זה כתבנו קוד נוסף הרי שאנחנו מעמיסים את המערכת מעבר למה שמצלמה מעמיסה; האם קריאת הסרטונים מהדיסק משנה את התנהגות המערכת כולה? ועוד ועוד.

מוסר השכל

אפקט הגשושית אינו קורה רק עם תוספת קוד או עומס. מתברר שהמון דברים יכולים להשפיע על תוצאות הבדיקות, וצריך להיות רגיש לאפשרות שהתוצאות משתנות עם שינויים בסביבה, או בגלל נתוני הסביבה; או בגלל הבודקים; או השעה; או הטמפרטורה; וכו' וכו'. זה המקום לציין שהסיכוי להשפעה כזאת גבוה יותר במדידות של ערכים רציפים. זו אחת הסיבות שבהגדרת דרישות לא-פונקציונאליות חייבים, כחלק מהגדרת הדרישה, לציין בפירוט איך תעשה הבדיקה, שכן שיטת הבדיקה משפיעה ישירות על הערכים הנמדדים ומכאן גם על התוצאה הסופית (עבר או נכשל) של הבדיקה.

איך עושים צעדים ראשונים - Playwright

זו כבר עובדה קיימת, יש לנו שחקן חדש ומבטיח בשוק של האוטומציה. מי שאפילו ירפרף בצורה הכי מהירה על קבוצות של רשתות חברתיות - יראה יותר ויותר שאלות/תשובות או אזכורים לגבי playwright. איך אפשר שלא? עם כל גרסא חדשה, הוא מציע יותר מדי דברים בתוכו.

קצת לפני שדוהרים קדימה

בין אם מתחילים פרויקט אוטומציה ממש ממש מהתחלה או בין אם מחליפים אחד קיים או אולי בכלל מוסיפים עוד אחד - חשוב לא לרוץ, במיוחד לא אחרי טרנדים, ולעצור לרגע ולהבין את הצרכים שלכם לטווח קצר וגם לטווח הארוך. יש לא מעט שאלות שכדאי לשאול לפני שבוחרים כלי / סיפריה / פריימורק לאוטומציה (לנו הייתה רשימה עם משהו כמו 100 שאלות שניסינו לענות עליהן תוך כדי בחינת כמה כלים).

כמובן שהשאלה הראשונה שנשאלת היא - מה עדיף לבחור? ועל זה אפשר לפתח דיונים שיימשכו ימים ולילות. במאמר הזה אנחנו נניח שבחנו כבר את הפריימורק שלנו, ושבחרנו ב-playwright. ואז נעבור לצמד שאלות שחוזר על עצמו לא מעט. איך מתחילים צעדים ראשונים עם הפריימורק ואיזו שפה כדאי לבחור (במידה והפריימורק תומך בכמה שפות).

על שאלות אלו אני רוצה לנסות לענות במאמר זה, על בסיס הדרך שעשינו עם playwright בחברת SeatGeek, בה אני עובד כיום. אבל לפני זה, קצת היסטוריה על תחילת הדרך שלנו.

התחלנו לעבוד עם playwright עוד בגרסאותיו הראשונות. כשזה עוד היה פשוט האח הצעיר של פריימורק אחר - puppeteer. אפילו אתר נורמלי לא היה ל-playwright להציע. אבל איכשהו בהסתמך על האינטואיציה החלטנו ללכת עם האח הצעיר ודי מהר התחלנו להתלהב מכל השיפורים ופיצ׳רים מעניינים שהיינו מקבלים גרסה אחר גרסה.

בימינו אנו (ינואר 2023) הוא יותר ויותר בשימוש ועומד על רף של יותר מ 1.4 מיליון הורדות שבועיות! יש כבר קורסים באתר יודמי. יש כבר סרטונים ופלייליסטים באתר יוטיוב, ביניהם גם הערוץ הרשמי של הצוות בו הם מעלים סרטונים עם ההסברים על כל גרסה שיוצאת.

גם במקרה הפרטי שלנו, כשאנחנו התחלנו את העבודה - לא היה לנו יותר מדי הכוונה. ניזונו מכל פיסת מידע קטנה על הפריימורק הזה. העבודה לא הייתה פשוטה. לא הכל עבד, לא הכל היה יציב. אבל לא ויתרנו על הרעיון של השימוש ב-playwright ולאט לאט הדברים התחילו להסתדר. התחלנו לצבור ניסיון מעשי בעצמנו וגם כן יצרנו מאגר ידע מטורף משלנו.

אחרי הקדמה קצרה, בואו וניגש לענות על השאלות שלנו.

שפת הפיתוח

נתחיל מזה שבחירת שפה יכולה להיות תלויה בכמה גורמים. אחד הגורמים שיש להם משקל מאוד גדול זה השפה שקהל המשתמשים (בין אם נוכחי, בין אם עתידי) שולט בה. אפשר לראות יותר ויותר שאת הטסטים האוטומטיים כותבים לא רק אנשי QA אלא גם מפתחים. בהרבה מקרים השפה של האוטומציה נגזרת מהשפה בה כותבים צוותי פיתוח. למשל, אם משתמשים בפייתון - הנטייה תהיה פייתון. אם אני אביא את הדוגמא של חברת SeatGeek - לפני תחילת העבודה עם playwright, צוות QA היו כותבים אוטומציה באמצעות סלניום בשפת Java אבל צוותי פיתוח שלנו משתמשים ב-TypeScript וב-C#. לכן כאשר ניגשנו לשאלה של שפת הפיתוח עם Playwright, עמדו בפנינו שתי אופציות של שפת הפיתוח:

- TypeScript - הינה השפה העיקרית בה משתמשים צוותי פיתוח.
- Java - הינה השפה שבה שולטים אנשי QA שלנו על בסיס עבודה של כמעט שנתיים עם פרויקט האוטומציה הקיים.

לאחר לא מעט מחשבה החלטנו שיותר קל יהיה להעביר את צוות QA ל-TypeScript (מדובר היה על 16 אנשים) וזה אל מול אופציה להעביר את אנשי פיתוח (בערך 50 אנשים) ל-Java. לעקומת הלמידה ישנו מחיר לא קטן של זמן וצריך לקחת את זה בחשבון.

לא תיארנו לעצמנו בתחילת הדרך - מה תהיה המשמעות של בחירת TypeScript, כי כאמור Playwright היה עוד בתחילת דרכו. עם הזמן, הבנו שהבחירה הייתה מאוד נכונה ולא רק בגלל שצוותי פיתוח שלנו משתמשים ב-TypeScript.

לאחר עבודה של שנה וחצי עם פריימורק, ההמלצה שלי היא אם ישנה אפשרות תלכו על האופציה של Playwright עם שפת JS\TS.

אחד הדברים הבולטים בצמד שכזה הוא העובדה שצוות של Playwright מפתח וממשמש את הפתרונות שלו, ללא תלות בפתרונות של צד שלישי. בתצורה הזו מקבלים (נכון לגרסה 1.30.0 של היום):

- דוחות מובנים של הפריימורק
- Test Runner מובנה של הפריימורק
- התקנה מהירה ע״י שימוש בפקודה אחת פשוטה

תוסף מטורף ורשמי של הצוות ל-Visual Studio Code, שכולל repository בנפרד שמאפשר לפתוח באגים לצוות.

אם להציץ על חיבורים עם שפה אחרת, למשל על Java, הכל יעבוד בצורה מעולה! זאת אומרת, הפונקציונליות של עבודה עם אלמנטים תהיה זהה. אבל אז אנחנו מגיעים לחלק של הרצת הטסטים. במקרה של Playwright + JS/TS - מקבלים את כל המכלול מהצד של צוות של פלייריט. הכל מפותח ומתוחזק ע"י הצוות עצמו. אם נתבונן בפתרון של חיבור עם שפת Java, אז כבר עושים שימוש (למשל) ב-TestNG בשביל הרצת הטסטים וזה פתרון מעולה. אבל יכול להיווצר מצב תיאורטי שמשהו ב-TestNG עצמו לא יעבוד כראוי, וזה יכול להפריע להרצה תקינה של טסטים, ואפילו בואו נניח שמדובר בבאג שצריך לתקן אותו. בגלל ש-TestNG זה לא הפיתוח של צוות Playwright עצמו - זה יכול לקחת יותר זמן עד שהבאג יטופל, מאחר ויש פה עוד צד נוסף שאמור לקחת חלק בתיקון.

והסנכרון לא תמיד עובד בצורה טובה ומהירה. לעומת זאת, כאשר כל הפתרונות מגיעים מבית יוצרים אחד, זה מקל על איתור הבעיה והפתרון שלה, וגם פה אני מדבר מניסיון, אחרי לא מעט באגים שהצוות שלי פתח.

אבל כאמור לשמחתנו, בלי קשר לשפת פיתוח שבוחרים - חשוב שתזכרו, שהרוב הגדול של פונקציונאליות הוא זהה לחלוטין. הטסטים קלים לפיתוח ועובדים מאוד מאוד מהר בכל אחת מהאופציות שתבחרו.

בנוסף, הצוות של Playwright באמת מקשיב לקהילה שלו ומנסה לתת פתרון לכל בעיה או הצעה לשיפור.

ועוד דבר - הצוות של Playwright מסתכל על דברים בצורה פרקטית ולהפתעתי כל פעם מחדש הם משחררים מגרסה לגרסה פיצ׳רים סופר שימושיים שפשוט כיף להשתמש בהם! ואני רוצה לחזור עוד הפעם על הנקודה הכי חשובה בעניין - הם מקשיבים לקהילת המשתמשים, ולא מעט שיפורים באים בזכות הבקשות של הקהילה עצמה.

הצעדים הראשונים

כאמור, אנחנו התחלנו את הצעדים הראשונים שלנו עם הפריימורק וללא ליווי של איזה מאגר מידע. הכל היה די בתחילת דרכו. היום, המצב הוא אחר לגמרי. יש המון מקורות מידע שאפשר לעבוד איתם.

אלכס קומנוב

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

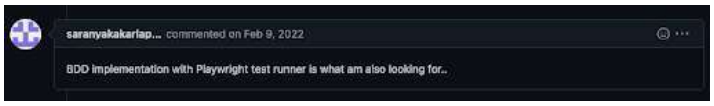
איך עושים צעדים ראשונים - Playwright

נתחיל מהדבר העיקרי - האתר הרשמי עם הדוקומנטציה הרשמית. אני יכול להגיד לכם בלב שלם שהאתר שלהם הוא אחד הדברים המטורפים! מאוד מפורט, עם המון דוגמאות קוד, שאפשר לקחת את התיעוד, להסתכל, להבין ולהעתיק איזו דוגמא ולממש אותה בחלק כזה או אחר בפרויקט. צוות של Playwright לא מתעצל להוסיף את מה שחסר לקהילה. הנה לכם לינק לבאג שאני פתחתי, כאשר זיהיתי חוסר בתיעוד של התקנת הדפדפנים, והתיקון של הבאג (בעצם הוספה של תיעוד) עתיד להיכנס באחת הגרסאות הבאות!

זה לוקח אותנו למקור מידע נוסף ומאוד חשוב, והוא ה-GitHub של הפריימורק. אפשר למצוא שם המון באגים שנפתחו, שאלות שנשאלו והצעות שיפור שהועלו. כל הדברים האלה נקראים issues ואם נעבור עליהם - אפשר למצוא המון פתרונות לבעיות של אנשי הקהילה, שדי בטוח שתיתקלו לפחות בחלק מהן. צריך לציין פה משהו, מצד אחד מענה על בעיות של קהל משתמשים לא שונה מרעיון של פלטפורמות כמו StackOverflow, אבל היתרון העצום פה שמי שעונה לרוב זה אנשי פיתוח של הפריימורק עצמו. וזה משהו שהוא שווה זהב. לקבל מענה לבעיה או לשאלה מצוות הפיתוח עצמו, זה משהו אחר לחלוטין. עוד משהו שאפשר למצוא ב-GitHub זה המון הצעות לייעול שאפשר לתת להם קול משלכם וככה לתרום לפיתוח מהיר יותר של פיצ'ר העתידי. עושים את זה באמצעות שני דברים פשוטים:

1. נותנים בלייק

2. כותבים תגובה שאתם הייתם רוצים גם כן לראות את הפיצ'ר הזה בעתיד



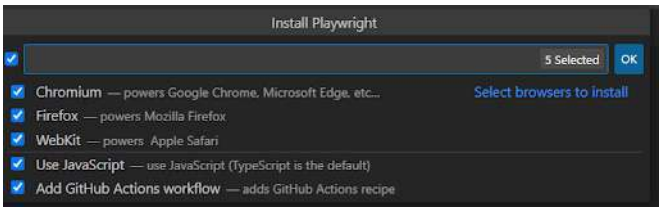
כאן מאוד חשוב לציין שהתקשורת עם הצוות והקהילה נותן תחושה מטורפת כפליים. בין אם מצאתם באג והצוות תיקן אותו, בין אם הבאתם הצעת ייעול שנכנסה כפיצ'ר, הרי עזרתם לשפר ולפתח את הפרויקט.

מקור נוסף ומאוד נגיש זה כמובן YouTube. מעבר לערוץ הרשמי של הצוות, אפשר בהרצת חיפוש פשוטה ולמצוא המון ערוצים טובים. למשל הערוץ הזה הוא מדהים בעיניי ואני לומד ממנו המון (וחייב לציין שהוא עזר לי המון בהתחלה).

עוד מקור שאני ממש ממליץ עליו זה קורסים של פלטפורמת Udemy. תמורת עשרות שקלים בודדים - אפשר לרכוש קורס שיתרום לכם המון!, הנה דוגמא לקורס עם java, קורס עם javascript, וקורס עם python.

אני רוצה להתעכב קצת על **<u>התקנה הראשונית</u>**. צוות של Playwright דאג לנו לשתי אופציות של התקנות סופר מהירות (כאשר בוחרים לעבוד עם JS/TS).

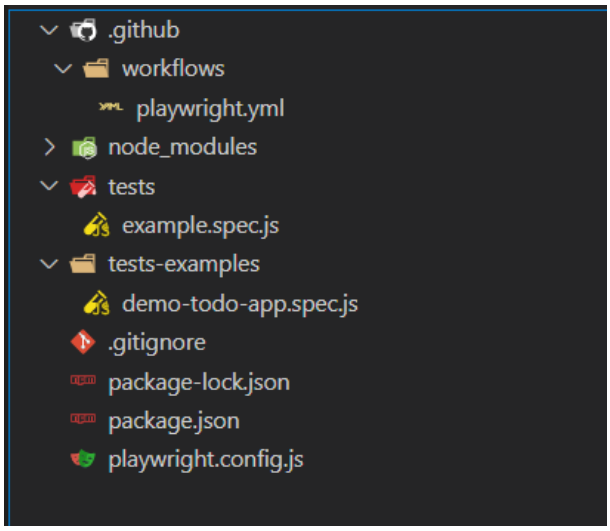
- התקנה עם שורת פקודה:
  - מריצים פקודה אחת פשוטה:
    npm init playwright@latest
  - בסרטון הזה אפשר לראות את ההתקנה הראשונית של הפרויקט ממש מתיקיה שהיא ריקה.
- התקנה דרך התוסף הרשמי של VSCode:
  - מתקינים את התוסף הרשמי
  - פותחים command palette
  - מריצים **Test: Install Playwright**
  - בוחרים את האופציות הרצויות ולוחצים על OK.



◄ בסרטון הזה אפשר לראות את כל תהליך ההתקנה.

הקסם הוא שהכל ירוץ בצורה אוטומטית, מבלי שנצטרך לעשות משהו נוסף.

בסופה של כל אחת מהאופציות מקבלים לא רק את הספריות הנחוצות לעבודה עם פריימורק והדפדפנים, אלא מקבלים מבנה ראשוני של הפרויקט.



כמו כן, מקבלים כמה טסטים לדוגמא, קובץ קונפיגורציה שלא רק מלא בכל טוב, אלא גם מלא בהסברים לגבי אותם ערכי הקונפיגורציה. **דוגמא:**

```
/**
 * @see https://playwright.dev/docs/test-configuration
 * @type {import('@playwright/test').PlaywrightTestConfig}
 */
const config = {
  testDir: './tests',
  /* Maximum time one test can run for. */
  timeout: 30 * 1000,
  expect: {
    /**
     * Maximum time expect() should wait for the condition to be met.
     * For example in `await expect(locator).toHaveText();`
     */
    timeout: 5000
  },
  /* Run tests in files in parallel */
  fullyParallel: true,
  /* Fail the build on CI if you accidentally left test.only in the source code. */
  forbidOnly: !!process.env.CI,
  /* Retry on CI only */
  retries: process.env.CI ? 2 : 0,
  /* Opt out of parallel tests on CI. */
  workers: process.env.CI ? 1 : undefined,
  /* Reporter to use. See https://playwright.dev/docs/test-reporters */
  reporter: 'html',
  /* Shared settings for all the projects below. See https://playwright.dev/docs/api/class-testoptions. */
  use: {
    /* Maximum time each action such as `click()` can take. Defaults to 0 (no limit). */
    actionTimeout: 0,
    /* Base URL to use in actions like `await page.goto('/')`. */
    // baseURL: 'http://localhost:3000',

    /* Collect trace when retrying the failed test. See https://playwright.dev/docs/trace-viewer */
    trace: 'on-first-retry',
  },
```

לסיכום העניין - צוות של Playwright באמת עושה עבודה מטורפת ובכל תצורה שתבחרו, אתם תהנו מהקסם שתקבלו! לרגע לא הצטערנו שבזמנו הבאנו אמון בזה. ומי שרוצה קסם מיוחד שנותן קצת אקסטרות, מוזמן לפחות לבדוק את הצמד של Playwright + TS/JS.

מוזמנים לבקר גם בערוץ יוטיוב שלי ובפלטפורמה החינמית שבניתי.

עושים QA לקריירה | איילת מלמד כהן

QA עצמי בדרך ליעדי 2023

אם יצאתם משיחת המשוב, קיבלתם יעדים ולא ברור לכם בדיוק מה לעשות שונה מחר, הטור הזה בשבילכם.

בתקופה הזאת, ברוב הארגונים מקיימים שיחות משוב והערכה שנתיות. אוהבים אותן או לא (מהמרת שפחות), פגשתם או שתפגשו את המנהלים שלכם שם, ולצד סיכום של השנה, תסתכלו קדימה ותדברו על יעדי שיפור ו/או התפתחות להמשך השנה.

יעדים כמו "להיות יותר עצמאי", "להתחבר לביזנס", או "ליזום יותר ולקחת אחריות E2E" נשמעים קצת כמו סיסמאות. אם הם לא ברורים מעשית, הם עשויים לתסכל, לגרום לסטרס ולחוסר ודאות לגבי היכולת להגיע אליהם בפועל.

יעדים מקבלים תוקף כשאנו מתרגמים אותם לפעולות קונקרטיות. בדיוק **כפי שאנו מתורגלים לעשות מול דרישות מוצר, נתרגם את יעדי השיפור וההתפתחות, פרקטית, כאילו היינו צריכים לכתוב להם בדיקות.**

לדוגמא: אתה QA בסקראם ויעד השיפור שהוצב לך הוא להיות "יותר עצמאי". ראשית, וודאו שאתם מבינים מהם הקריטריונים להשגת היעד הזה ולמה אתם זקוקים? השתמשו בשאלות האלו (*ובתשובות לדוגמא*) ובדקו:

1. **איך נדע, אני והמנהל.ת שלי, שהגעתי אל היעד?**

זמן הליווי שאני זקוק מהמפתחים בצוות יפתח, פידבקים מחברי הצוות, אנהל קובץ מקיף של כל ה-Tricks&Tips הנחוצים סביב הקמת הסביבה והרצת הבדיקות

2. **אילו התנהגויות יאפיינו אותי כשאהיה יותר עצמאי.ת?**

אפתור יותר בעיות לבד, אדע למי לפנות בכל קונטקסט ולא אזדקק לליווי צמוד בהקמת הסביבה

3. **מה נדרש ממני כדי להיות יותר עצמאי.ת?**

להשקיע יותר זמן בלפתור בעיות בכוחות עצמי, למפות לעצמי את סוגי האתגרים שהיום אני זקוק לליווי הדוק עבורם

4. **לאיזה סוג של תמיכה אני זקוק.ה כדי להצליח?**

מנטורינג על הקושי, זמן ללמוד Offline להשלמת הפערים שזיהיתי, סקירה של המפתחים בצוות על המדריך שאכתוב

לפרט את היעד בצורה כזאת דומה להגדרת ה-**Expected results** בתרחיש בדיקות ואת **ה-Prerequisites** לכך שהתרחיש יוכל לרוץ והתוצאות יתממשו.

הצלחתם לתרגם בצורה כזאת את היעדים שקיבלתם בשיחת ההערכה שלכם? אם כן, מעולה! מוכח שההסתברות לכך שהם אכן יתממשו, זה עתה עלתה ב- 150%!

אם לא, שתפו את המנהלים שלכם, שנחוצה לכם בהירות כלפי הדרך בה היעדים יתממשו בפועל, ובקשו להגדיר כלי מדידה יחד. בכך שאתם עושים את זה, אתם לוקחים אחריות מלאה על היעדים וממקדים את האנרגיות שלכם במימוש יעיל שלהם.

אתם לא מרגישים בנוח לחזור ולבקש מהם את זה? העובדה הבאה בשבילכם:

הפרויקט המחקרי של גוגל, Oxygen, בדק מה מאפיין את המנהלים הטובים ביותר?

נמצאו 10 תכונות ומיומנויות, ובמקום הראשון: המנהל כמאמן. כלומר, פעיל כשותף בפיתוח של חברי צוותו ומימוש יכולותיהם. ממצאים אלו קיבלו חיזוק משמעותי גם במסקנות מחקרי הענק של חברת האנליטיקס האמריקאית, Gallup, שמצאו שיכולות אלו בדיוק, גם מנבאות ביצועים גבוהים יותר לארגונים בטווח הארוך.

> **"כפי שאנו מתורגלים לעשות מול דרישות מוצר, נתרגם את יעדי השיפור וההתפתחות, פרקטית, כאילו היינו צריכים לכתוב להן בדיקות."**

אז אם המיומנות הכי נדרשת למנהלים בעולם העבודה החדש היא לחנוך ולאמן את חברי הצוות שלהם, בכך שאתם חוזרים אליהם בשאלות האלו אתם בעצם עוזרים להם לממש כישורים אלו ולהיות מנהלים טובים יותר!

אם זכיתם במנהלים שאתם מרגישים שהם מנטורים ומאמנים, אתם בוודאי מרגישים את השותפות שלהם באתגרי השיפור וצמיחה שלכם. ההקשבה, השאלות שמאתגרות אתכם לחשוב מעבר ל"מה שצריך לעשות" והליווי שלהם שתומך במטרות הארגון, **לצד המטרות שלכם,** מקדם את היעדים של שניכם.

אם לא, No Worries, יש עוד הרבה מה לעשות בעצמכם ולמענכם.

תתחילו בשאלות שהוזכרו כלפי היעדים והמשיכו לדרך בה המנהל שלכם ילווה את מימושם בפגישות האישיות איתכם.

מנהלים שממממשים מיומנויות אימון וחניכה, יכירו את החוזקות שלכם, ויעזרו לכם להשתמש בהם כדי להתקדם אל עבר מימוש היעדים שלכם. הם לא רק יעקבו אחר היעדים ויתנו לכם פידבק אם הגעתם אליהם או לא, אלא יהיו ה-Accountability partners שלכם בדרך לשם.

כיצד מיושמת השותפות הזו?

אחרי שישנה הבנה והסכמה לגבי היעדים והמדדים להשגתם, ואתם יודעים לענות על השאלות שהוזכרו, גזרו מהם תוכנית פעולה אל עבר ה-Expected results. דמיינו תוכנית לכל יעד, ברזולוציה של חודשים, ובכל חודש, אזור פוקוס שמפרק את היעד לפעולות מוחשיות בהקשר של אותו יעד. מעין אבני דרך בדרך למימושו.

בקשו להיפגש עם המנהל.ת שלכם אחת לחודש כדי ללוות אתכם בתוכנית הזאת.

למשל, ביעד "להיות עצמאי" המיקוד של החודש הראשון, אבן הדרך הראשונה, יכול להיות, למשל, מיפוי אזורים / סוגי בעיות ואתגרים

איילת מלמד כהן

בעלת ניסיון של 18 שנה בעולם ה-QA, רוב השנים כמנהלת בכירה בסטארטאפים וחברות גדולות. בשנתיים האחרונות מאמנת מנהלים להתפתחות אישית ומקצועית, שיפור מיומנויות ניהול, כניסה לתפקיד חדש או אתגרי ההתמודדות עם הקיים. בנוסף לאימון, איילת מלווה סטארטאפים בכל הקשור לניהול איכות, בונה צוותים ואסטרטגיות QA מותאמות לארגון. בוגרת CTI, בעלת תואר ראשון בכלכלה ושני במנהל עסקים מאוניברסיטת בר-אילן. אמא לשלושה וזוקפת לזכותם חלק גדול ממיומנויות הניהול שלה.

עושים QA לקריירה | איילת מלמד כהן

בהם נחוץ ליווי. אחרי שלומדים מהמיפוי בשיח משותף עם המנהל.ת, בוחרים את המיקוד לחודש השני, כהמשך אליו לדוגמא, לקחת את הנושא הכי שכיח בו עד היום הייתה נחוצה עזרה וללמוד אותו.

הקריטריונים לאבן הדרך יהיו למשל לתעד Tricks&Tips שקשורים אליו, לעשות סקירה עם מי שצריך, לשמר את הידע וכל הנחוץ כדי להפוך את הידע זמין לכל מי שיצטרך את זה - בצוות או מחוצה לו. הקריטריונים האלה יהיו ה-Exit Criteria כדי לעבור לאבן דרך הבאה.

"עצם הפניית הקשב ותשומת הלב אל השותפות והיעדים, משדרת את האכפתיות והאחריות שלכם."

נצלו את המפגש האישי בכל חודש, או כל רזולוציה אחרת שהסכמתם עליה, כדי להיזכר ביעדים, ובתוכנית לגבי כל אחד מהם, ובדקו בשיח משותף: איך הולך? מה כן התקדם? מה למדתם מזה? מה דרוש כדי להמשיך להתקדם ואיזה פוקוס צריך להיות לחודש הזה? אם לא התקדמתם - יש מה ללמוד גם מזה.

גם אם יצרתם את התוכנית לבד והמנהלים שלכם לא שואלים אתכם את השאלות האלו, תעלו אותם אתם ובכך תרתמו אותם לתמוך בכם. תביאו ל-1:1 אחת לכמה שבועות את היעדים ושתפו: ב-X התקדמתי, ב-Y לא, ב-Z אני צריך/צריכה עזרה ואלו האתגרים שלי, ב-Y הייתי רוצה מנטורינג. אם אתם רוצים להתייעץ איתם כדי להתקדם יותר - הנה יצרתם הזדמנות מובנית לעשות זאת.

הלמידה מהדרך היא משותפת, **ועצם הפניית הקשב ותשומת הלב אל השותפות והיעדים, משדרת את האכפתיות והאחריות שלכם. היוזמה שלכם, תרתום את הצד השני להיות שותף פעיל לדרך ולהצלחה בה.** וברמה הכי פרקטית - כשאנו אומרים בבירור למה אנו זקוקים כדי להצליח, אנחנו מכפילים את הסיכוי לכך שזה יתממש.

בהצלחה לכולנו, גם ב- 2023!

מי עדיין לא שמע על ChatGPT

ה-ChatGPT הינו צ'ט מבוסס בינה מלאכותית שפותח ע"י חברת OpenAI (שאגב, מייקרוסופט מממנת 20% מהפרויקט). זהו מודל שפה (שימוש בסטטיסטיקות מבוסס מילים) שלומד ומשתפר עם הזמן (נכון לעכשיו עד 2021).

ישנם דיונים רבים ברשת על האם טכנולוגיה זו הולכת להחליף אותנו - בודקי התוכנה והאוטומציה, חלקינו מאמצים אותה, חלקינו נרתעים ממנה, אבל בהחלט לא ניתן להתעלם ממנה.

עם הזמן, ככל שאני לומד להכיר טוב יותר את מודל השפה הזה, אני מבין כי להחליף אותנו - אנשי הבדיקות הוא אינו יכול, לא היום בכל אופן.

בניגוד למכונות ואלגוריתמים תכנותיים למיניהם, בני האדם מביאים איתם חשיבה וזווית ראיה שונה, שזה כל כך חשוב בבואנו לבדוק תוצר של מישהו אחר (תוכנה \ חומרה או אפילו תסריט שמישהו כתב), אנחנו גם מתקשרים וורבלית עם אנשי הפיתוח (כל עוד הם לא גם כן בוטים למיניהם...), עושים סיעור מוחות על קפה במרפסת, בשיחות מסדרון עולים הרעיונות הטובים ביותר ובארוחת הצהריים לפעמים פותרים בעיה שמציקה לכל הצוות.

אז מה? ה-ChatGPT הינו סתם הייפ? ממש ממש לא!

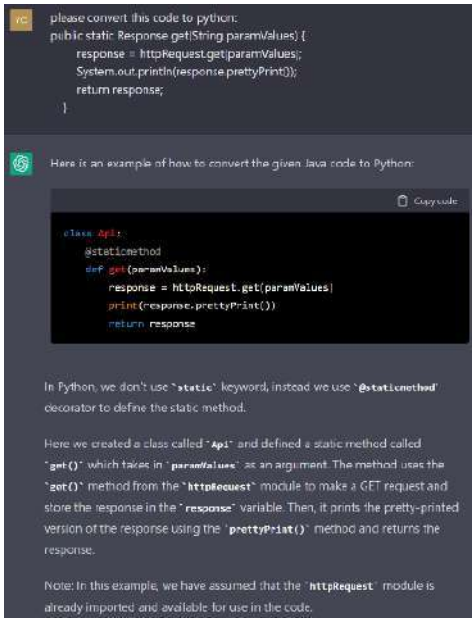
אני רואה את הטכנולוגיה הזו, בדיוק כמו שאר הטכנולוגיות וכלים אחרים בכך שהם באים לעזור לנו, אנשי ה-QA, לבצע את תפקידנו בצורה טובה יותר. עם ChatGPT נוכל לכתוב קוד אוטומציה (הרבה) יותר מהר, נוכל להבין מה זה Throughput בבדיקות עומסים, למצוא פונקציה שמביאה לנו נתונים מ-Data Set לבדיקות בצורה יעילה ועוד... בדיוק כמו שה-Wireshark הופך אותנו לבודקים טובים יותר, או ה-Jest או מה שזה לא יהיה.

תזכרו, טכנולוגיות לא מחליפות אנשים, אם כבר, ההיפך הוא הנכון (בזכות הסמארטפונים למשל, יש כיום תעשיית פיתוח אפליקציות שמעסיקה מיליוני אנשים).

כבן אדם סקרן שתמיד אוהב לחקור ולהתנסות בטכנולוגיות חדשות בתחומי (אוטומציה), החלטתי להפשיל שרוולים ולבדוק בעצמי במה ה-ChatGPT יכול לעזור לי, לאחר כמה משחקים ועבודה מצטברת של כמה שעות עם ה-ChatPGT, נוכחתי לראות את היתרונות הברורים בעבודה עם ה-Chat ולהלן 10 מהם:

1️⃣ כתבתי כמה מקרי בדיקה פשוטים וכאלו קצת יותר מורכבים עם Playwright, ביקשתי מה-ChatGPT לעשות את העבודה שאני הכי פחות אוהב לעשות בעבודת הפיתוח - לתעד את הבדיקות שלי, הוא עשה זאת בצורה מצוינת.

2️⃣ כתבתי פונקציה שעושה parsing לjson\מ , הקוד היה מעפן ולא יעיל, ביקשתי מה-ChatGPT לעשות לי Code Refactor, הוא לא אכזב, שיפר לי את הפונקציה בצורה מופלאה, קיצר אותה וכתב בצורה קצרה יותר ואינטליגנטית הרבה יותר.



3️⃣ הוא ידע להמיר לי פונקציות עזר (wrappers) ב-API משפת ג'אווה לשפת פייתון ולהיפך !

4️⃣ ביקשתי ממנו עזרה במימוש של Design Pattern מסוג Singleton על הדרייבר שלי, עשה את זה כמו גדול.

5️⃣ כתבתי ביטוי ב-RegEx כדי לתפוס תבניות של Errors או Warnings מתוך קובץ לוג, ה-ChatGPT שיפר לי את הביטוי, הוא גם יכתוב לי כל ביטוי שאתאר לו

6️⃣ כנ"ל לגבי שאילתות SQL

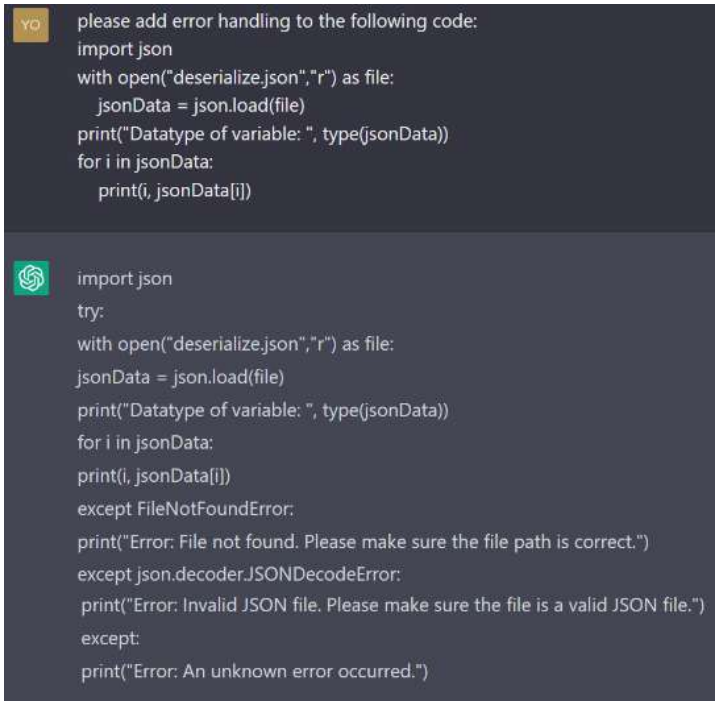
7️⃣ לקחתי קוד אוטומציה אקראי מ-GitHub, בכוונה חיפשתי משהו מסורבל (ניתן למצוא לא מעט כאלו), ביקשתי מ-ChatGPT שיסביר לי מה הקוד עושה. זה היה כמו לשמוע מוסיקה קלאסית - כל כך נעים לאוזן

8️⃣ השתמשתי לא נכון בסינטקס של Allure Report על כן הטסט לא דיווח, מה הבעיה ? יש ChatGPT שיפתור לי ת'פלונטר.

9️⃣ ב-Unit Test הוא בכלל מלך, נתתי לו פונקציה שיוצרת חיבור לשרת MySQL (עם הפרמטרים של הכתובת, השם משתמש והסיסמא), ה-CahtGPT כתב לי בדיקות יחידה על פונקציה זו (כולל תיעוד)

🔟 התותח הזה יודע גם לכתוב לי Dockerfile וגם פייפליינים בג'נקינס (בינתיים לא ניסיתי דברים מורכבים), בהינתן רשימה של צרכים מצידי.

לאנשים שעדיין סקפטיים או מצקצקים, יש לי חדשות טובות ורעות עבורכם:



הטובות: ה-ChatGPT גם פישל עם התשובות שלו, ולא מעט, צריך לבדוק אותו ולעשות לו QA

הרעות: שימו לב כי ה-ChatGPT בגרסתו הנוכחית (3.5) יצא בסה"כ לפני בערך חודשיים, תמתינו שנתיים ותראו לאן זה יוביל אותנו, אני זוכר שגם את האייפון הראשון (אי שם בשנת 2007) רבים קטלו, ותראו היום לאיזו מהפכה הוא הוביל...

חברים, החדשות "הרעות" הן בשורה של ממש וחדשות טובות לכולנו, המפתחים, הבודקים, המשתמשים - כולם. תשאפו לאמץ טכנולוגיות חדשות (ולבדוק אותן), בטח ובטח בתחום כמו שלנו אשר מתחדש כל הזמן.

בהצלחה

יוני

יוני פלנר

נשוי+4, גר בניצני עוז (כשהוא בארץ)
עובד בתחום הבדיקות (ידניות ואוטומציה) מ-2005, בשנת 2015 פתח את מכללת "עתיד האוטומציה" המתמחה בהדרכות | ייעוץ | הטמעה של פרויקטי אוטומציה מורכבים, כיום מנהל את החברה ומדריך ביחד עם סאיד ג'בר.
במסגרת עבודתי, חוקר כלים, מתודולוגיות וטכנולוגיות חדשות בתחום בדיקות התוכנה בכלל והאוטומציה בפרט.

משולחנו של שביט - מה הופך בודק בינוני לבודק גבוה?

בכתבה הזו אני רוצה לדבר אתכם על המעבר הזה בין בודק שנמצא בתחום וגם מודע לעצמו ומבין שהוא ברמה בינונית והוא חייב לשפר את יכולותיו כדי להגיע רחוק ולהתקדם בתפקידים או פשוט להתפתח ולעלות שלב. להפוך בעצם לבודק ברמה גבוהה יותר. כזה שכולם סומכים על דעתו המקצועית.

לא פעם נתקלנו במצבים בהם חשבנו שאנחנו מבינים הרבה מאוד, אבל אנשים מקצועיים מאיתנו ידעו לשים את האצבע ולגרום לנו להבין שיש לנו בעצם לא מעט לשפר ואפשרו לנו להבין שתמיד יש לאן לשאוף ושאנחנו צריכים לשמור על open-mind בגדול.

אבל בשביל זה צריך להשקיע זמן, להקריב, לעבור דרך בלשנות פאזה.

צריך ללמוד ולתרגל בכלים חדשים, שיטות שונות שאנחנו אולי מכירים ושמענו עליהם אבל לא בהכרח התנסנו וחווינו בעצמנו.

ישנם לא מעט בודקים כאלו שאחת הדרכים שלהם זה להפוך לבודקים עם יכולות פיתוח ואוטומציה, ישנם כאלו שזו המקפצה שלהם לניהול וישנם כאלו שלא מעוניינים בפוליטיקה וכל המסביב אבל עושים מה שהם עושים בצורה ממש טובה ויסודית עם מחשבה רחבה ביותר והם מאוד mind set לכל נושא ה – Quality.

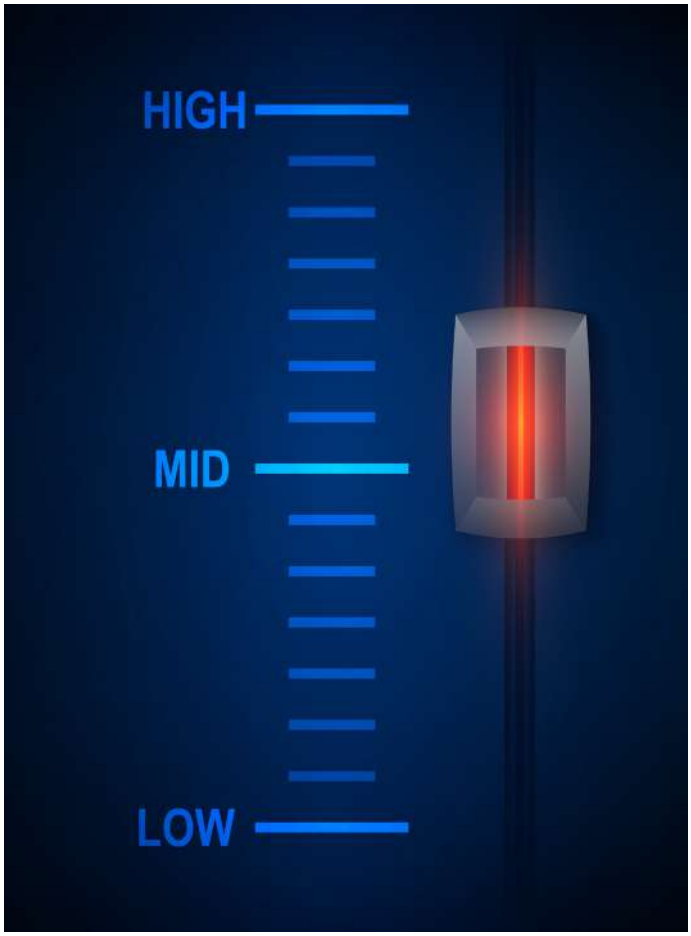
מגיע השלב שאנחנו לא יכולים להישאר באותו mode כמו שאנחנו רגילים. להיכנס לפיצ'ר, לעבור על האפיון, לכתוב מסמך, לבדוק, לסמן תוצאות, לתעד באגים, לבדוק בדיקה חוזרת ולתת אור ירוק שהכל תקין ואפשר להתקדם לייצור.

מגיע שלב שבו אנחנו חייבים לקחת יוזמה, להראות את המסוגלות שלנו לשנות את ההתנהלות הדיפולטיבית שלנו ואת מה שהיינו מורגלים לעשות בכל סיטואציה.

עצה שלי, אל תתרגלו ותרגילו את עצמכם לשגרה.

כשיש לכם יום שהכל עובד, הכל רץ, הכל דופק, הכל במקום ואין בעיות - אל תנוחו על זרי הדפנה ותהיו רגועים מדי.

תמיד תשאלו את עצמכם, מה אני יכול לעשות טוב יותר?, איך אני יכול לשפר את זה? ותמיד תתהו אם אתם לא מחמיצים משהו.



אז מה עושים ? איך מתקדמים ויוצאים מאזור הנוחות ו-עולים שלב?

1. תחקרו כל פעם עולם אחר ותלמדו לעשות דברים שאתם לא יודעים. עושים רק UI? – תלמדו API. לא בודקים מספיק DATA? – תלמדו ותחקרו את התחום ואלו בדיקות DATA שאפשר לעשות.
2. תביאו אל שולחן של המנהלים שלכם יוזמות חדשות. תראו שאכפת לכם וחשוב לכם לשפר את כל נושא ה-QUALITY.
3. תהיו ביקורתיים כלפי עצמכם – על כל טעות משמעותית – בצעו תחקיר מסודר עם מסמך סיכום שנקרא Lesson learned וציינו את הנקודות הקריטיות שהייתם צריכים לתת להם יותר תשומת לב ולהיות ערניים יותר.
4. מוניטורינג ו – על כל דבר שאתם עושים. אוטומציה, קצב עבודה, כיסוי מלא ו review של טסטים ברמה גבוהה וללא פשרות.
5. דוחות של טסטים, דוחות של כיסוי טסטים, ידניים ואוטומטיים.
6. מיילים על עדכונים שבועיים או דו-שבועיים מצוות ה-QA בצירוף דוחות והצגת מספרים החוצה לכלל החברה.

אל תהיו במחשכים! כולם כולם צריכים לדעת מה בדיוק אתם עושים, מה ההתקדמות שלכם ואיך ומה בדיוק אתם מקדמים. תטביעו חותם בחברה ותוודאו שכולם יודעים מה אתם עושים.

הסינים אומרים -

"הזמן הכי טוב לזרוע עץ הוא לפני 20 שנה. הזמן השני הכי טוב – הוא עכשיו!"

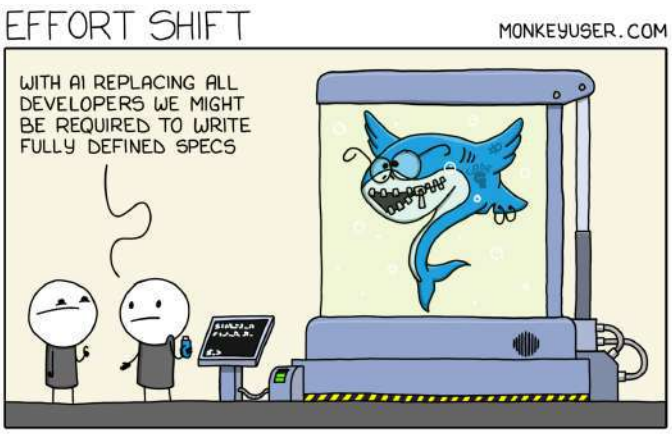
שיהיה לכם המשך שבוע טוב

ושלעולם לא תצעדו לבד

שביט ג'רסי

מנהל בדיקות בחברת Wisestamp, בעל תואר ראשון בהנדסת תעשייה וניהול מהטכניון, בעל 10 שנות נסיון בתחום הבדיקות, מתוכם מעל 6 שנים מדריך עצמאי ל-QA.

בעל סדרת הסרטונים השבועית "QA ללא הפסקה"



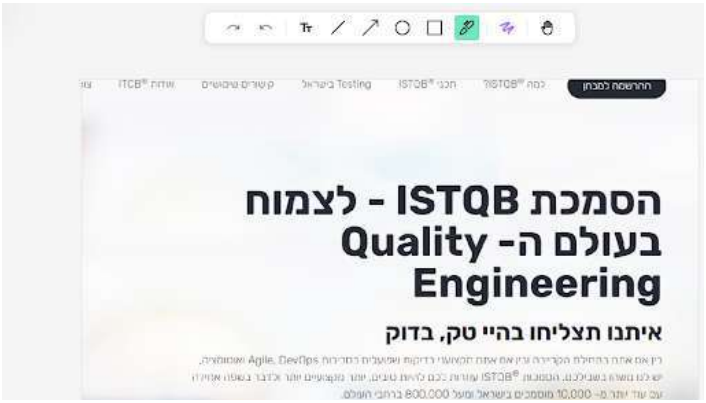
סקירת כלים Jam.Dev – ניצן גולדנברג

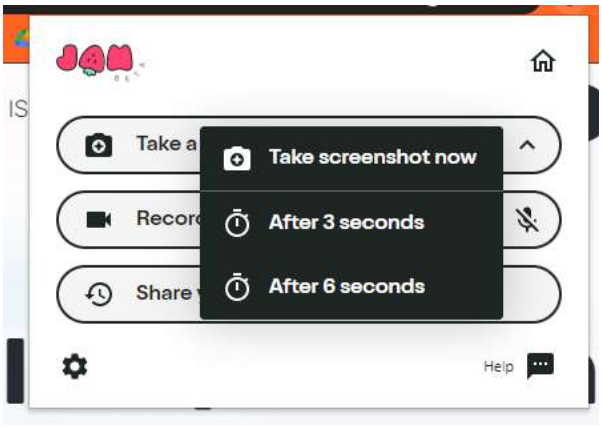
בשנה האחרונה נכנס לשוק כלי חדש אשר מראה יכולות מעניינות שיכולות להקל מעט את העבודה של הבודקים. לכלי קוראים Jam.Dev והוא בנוי כתוסף לדפדפנים.
ואלו הן היכולות של הכלי:

- צילום מסך
- צילום סרטון וידאו
- תפיסת וידאו של ה-3 דקות האחרונות
- שמירת נתונים בענן
- מידע חשוב מהדפדפן
- התממשקות
- קבוצת עבודה (שדרוג החשבון)

צילום מסך

Jam מאפשר לצלם את המסך ע"י צילום יחיד (Screen Shot) ולערוך את התמונה לפני שמירתה. כמו כן ניתן לבחור אפשרות של צילום מסך עם טיימר של 3 ו-6 שניות.

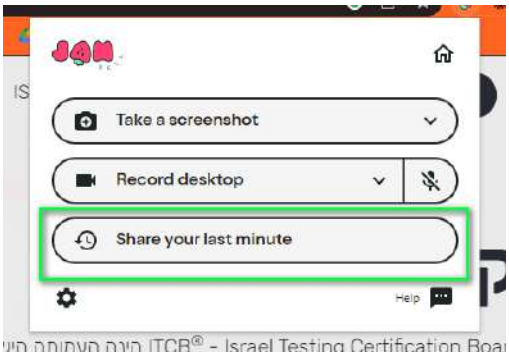




תפיסת וידאו של שלושת הדקות האחרונות

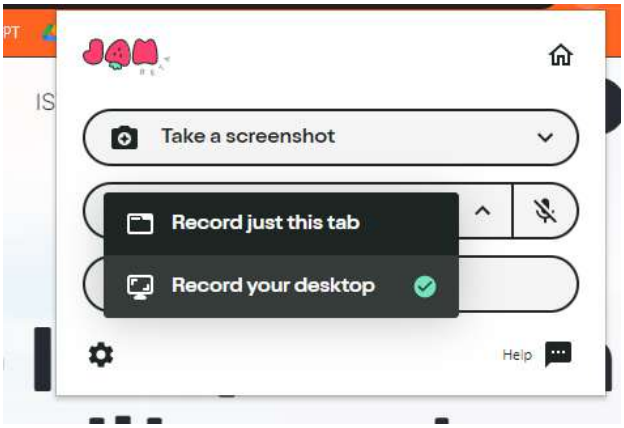
ל-Jam היכולת לשמור את 3 הדקות האחרונות של השימוש במסך ע"י שימוש בלולאה לא נגמרת אשר נשמרת בענן של הכלי. היכולת הזו מתאימה מאוד לבאגים אשר "פספסנו" ולא הספקנו לתעד או לא הצלחנו לשחזר (וכולנו יודעים שגם באג שמשתחזר בפעם אחת עדיין יחשב לבאג!!)

Jam יתעד רק תזוזות שקורות בדפדפן, כלומר, אם לא הזזנו את העכבר או עשינו פעולה מסוימת בדפדפן אז הוא יודע שאין צורך לתעד.



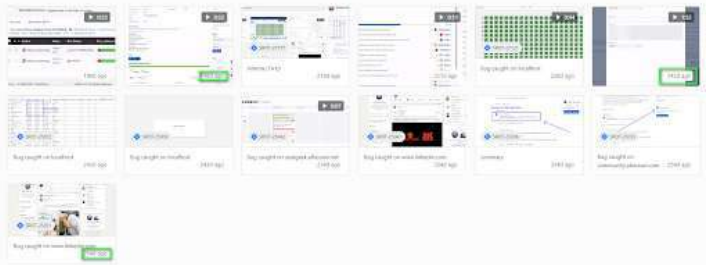
צילום וידאו

Jam מאפשר ליצור סרטוני וידאו הן של הדפדפן בלבד והן של שולחן העבודה (ניתן לאפיין לפי עדיפות). כמו כן, ניתן לאפיין אם רוצים להוסיף קול לסרטון וניתן גם להגדיר סרטון ללא קול.



שמירת נתונים בענן

כל צילומי המסך והסרטונים נשמרים בענן של Jam וזמינים לשימוש ללא הגבלה של זמן או מקום.



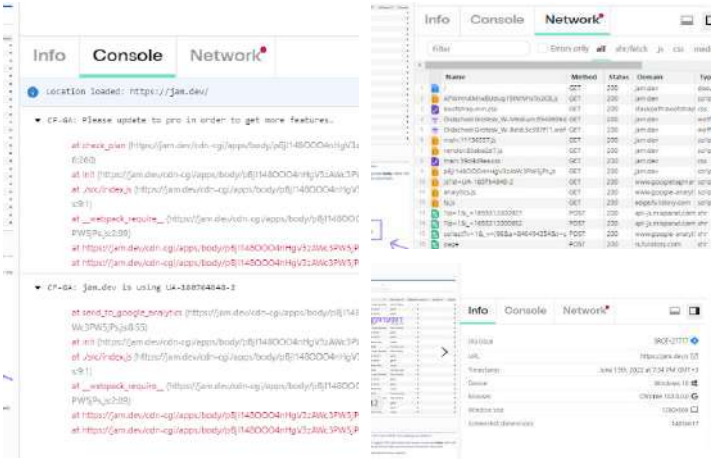
ניצן גולדנברג

מזה 7 שנים בתחום בדיקות התוכנה, תפקיד נוכחי מהנדס בדיקות בכיר בחברת SeatGeek, מנהל קבוצת ה-AB של ארגון ISTQB® המוביל הראשי של קבוצת המיטאפ TestIL, מרצה בקורסים לבודקי תוכנה

סקירת כלים Jam.Dev – ניצן גולדנברג

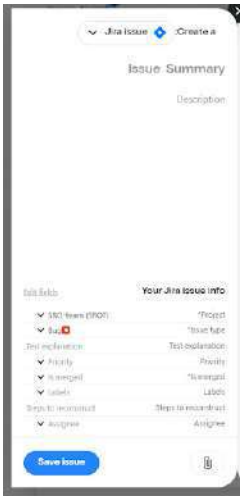
מידע חשוב מהדפדפן

בעת יצירת באג חדש הן ע"י צילום מסך והן ע"י סרטון וידאו Jam אוספת נתונים מהדפדפן כגון כתובת האתר, זמן לקיחת הצילום, מאיזה מכשיר וגרסת הדפדפן הצילום נלקח, רזולוציית המסך, מידע מהקונסול של הדפדפן, ופרטי תעבורת הנתונים. המידע הזה מצורף לדיווח של הבאג



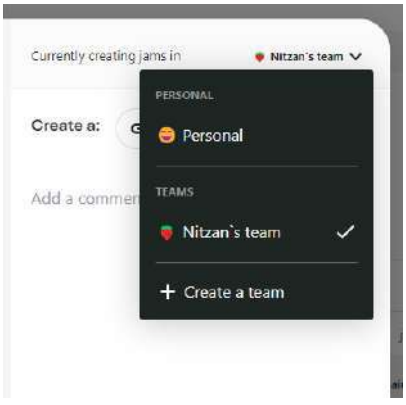
התממשקות

ל-Jam היכולת להתממשק עם כלים חיצוניים לדיווח באגים כגון: ClickUp & Jira, Linear, Asana Slack, GitHub, כך שברגע שעושים צילום מסך או וידאו, ניתן מיידית לפתוח באג במערכת המיועדת ללא צורך להיכנס במיוחד למערכת.



ברגע שבחרנו את המערכת המיועדת, יפתחו לנו כל השדות לדיווח הבאג כפי שהשדות מוצגות ישירות במערכת עצמה (כותרת הבאג, צעדים לשחזור, תוצאה צפויה ותוצאה בפועל, חומרה, תיעדוף או כל שדה אחר שמאופיין לנו במערכת)

קבוצת עבודה



Jam מאפשר ליצור קבוצת עבודה ולשתף את צילומי המסך והוידאו עם כל חברי הצוות ללא הצורך לשלוח קישור ישיר.

ניתן לאפיין אפשרות התחברות לקבוצת עבודה ע"י מייל יחודי (כגון מייל ארגוני) או כל מייל אפשרי.

בעת לקיחת צילום המסך או הוידאו, ניתן לבחור אם רוצים שהצילומים ישמרו בענן פרטי או בענן של קבוצת העבודה מה שאומר שכל המידע שצולם יהיה זמין לכל חברי הצוות.

יתרונות

- תוסף דפדפן אשר אינו מצריך התקנה מיוחדת
- קינפוג נוח וקל של ההתממשקויות
- ממשק משתמש נוח וקל לתפעול
- שרותי ענן ללא הגבלת נפח
- אפשרות לתפוס את השימוש האחרון של הדפדפן
- יצירת צוות ושיתוף מסמכים בצורה פשוטה ונוחה (בתשלום נוסף)
- לכל צילום מסך או וידאו מתווסף מידע רלוונטי מהדפדפן
- היכולת ליצור קבוצת עבודה ולשתף צילומי מסך ווידאו עם חברי הצוות (בתשלום נוסף)
- אוזן קשבת לבעיות ע"י בעלי החברה (כולל עדכון על סטטוס התיקון)

חסרונות

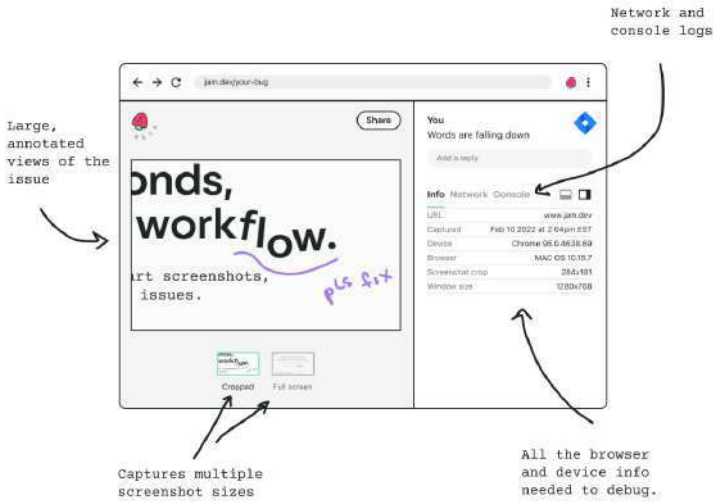
- עדיין נחשב כגרסת Beta
- אין קהילת משתמשים
- עובד רק על דפדפן כרום

לסיכום

Jam עדיין נחשב כגרסת Beta אך הוא מראה פוטנציאל אדיר להיות כלי מרכזי בחיי הבודקים והמפתחים. למרות שיש כלים רבים בשוק כיום אשר מאפשרים צילום מסך או וידאו, עדיין Jam מוסיף ערך נוסף מהותי של מידע לגבי השימוש בדפדפן והתממשקות עם מערכת ניהול התקלות, דבר אשר חוסך לנו שימוש בכלים נוספים.

אני יכול לומר לכם שאצלי בארגון הבודקים והמפתחים משתמשים ב-Jam בעבודה לפתיחת תקלות וזה חוסך להם הרבה זמן יקר.

ציון
מענה לצרכים של החברה 10/10
נותן למשתמש חווית שימוש 10/10
תמיכה וקהילה 4/10
סה"כ 8



בדיקות API אוטומטיות עם אינטגרציה לסלניום | נתי צדוק

API הם ראשי התיבות של Application Programming Interface, הפועל כממשק בין שני יישומים ומעניק יכולת לתקשר זה עם זה ללא קשר לשפות התכנות המשמשות לפיתוחם.
בדיקת API ממלאת תפקיד מכריע בתהליך פיתוח התוכנה שכן היא עוזרת לנו לבדוק את ההיגיון העסקי של האפליקציה עוד לפני שה-UI מוכן. שכן, ממשק API שלא כמו כל היבט אחר של התוכנה הוא חסר GUI אך ההשפעה שלו ניכרת בכל התוכנה.
עבור קהילת הבודקים, בדיקת API Automation מהווה נישה חדשה, המורכבות של קבצי JSON גורמת לבדיקות API לא להיות מנוהלות בקוד, ולכן נראה פחות צוותים המנהלים קוד עבור זה.
מכאן, שזה המקום שבו אוטומציה נכנסת פנימה ומסייעת בבדיקת ממשקי API, כל מימוש הקריאות/ בקשות נכתבות פעם אחת לשימוש חוזר עתידי והתשתית בעצם נכתבת פעם אחת ומשמשת עבור כל API חדש שנרצה לממש עבורו בדיקה.

נתי צדוק

מפתחת תשתיות אוטומציה, עובדת כ-8 שנים בחברת Bmc Software
בעלת ידע נרחב בבדיקות תוכנה וכתיבת אוטומציה UI, Backend ו-API
בעלת תואר ראשון במדעי המחשב ומתמטיקה בעבר מרצה לשפות תכנות במכללה הטכנולוגית "תל-חי"
מנהלת סניף בארגון she codes

ישנם המון כלים עבור אוטומציה ל-API ואחד המוכרים והשימושיים כיום הוא Postman REST Assured היא אחת מספריות ה-Java הפופולריות ביותר המשמשות לכתיבת בדיקות ולניטור של בדיקות עבור שירותי האינטרנט RESTful. ישנם סוגים שונים של ממשקי API כגון SOAP, REST ו-RPC וכו'. אני אתמקד בבדיקת REST API באמצעות REST Assured .

הסיבה שבגינה אתמקד ב-REST API באמצעות REST Assured, היא:

- קוד פתוח, מבוסס Java, ותומך הן בפורמט XML והן בפורמט JSON.
- משתלב היטב עם Maven, TestNG.
- תמיכה ב-DELETE, POST, GET, PUT, PATCH ו-OPTIONS ובכל הבקשות העיקריות.
- אימות התגובה שהתקבלה בהתבסס על Groovy ופילטור מתקדם בעזרת Gpath
- לספריה ישנן שיטות להביא נתונים כמעט מכל חלק של הבקשה והתגובה, ללא תלות במורכבות מבני ה-JSON.
- לצד התמיכה בכל שיטות ה-HTTP, הוא תומך גם בבקשות הכוללות OAuth ו-OAuth 2.0.

ישנן סיבות רבות מדוע מומלץ לכתוב בדיקה אוטומטית לרמת ה-API, אך הסיבה הטובה ביותר תהיה מהירות הביצוע המוגברת של בדיקות אלו בהשוואה לזו של בדיקות מבוססות GUI, יחד עם היקף הרחב יותר.

החלטתי לאמץ את התשתית לצוות שלי בזכות החיבוריות המאוד טובה עם סלניום.

פורמט של פונקציה

איך בעצם טסטים נכתבים ב-REST assured

(Behavior Driven Development) BDD

העיקרון של בדיקת BDD הוא מקרי המבחן הנכתבים בשפה טבעית ניתנת לקריאה בקלות גם על ידי מי שאינם מתכנתים.

כאשר נשלח את הפרמטרים הנכונים - Given

- ונבצע מתודה מסויימת When

- נצפה ל.. Then

```
public void Get(String URL)
{
        RequestSpecification httpRequest =
        .given()
              .header("x","y")
        .when()
        .get(URL)
        .then()
              .statusCode(HttpStatus.SC_OK)
        .extract()
        .response();
}
```

בניית תשתית הפרויקט

נבנה פרויקט Maven בשימוש TestNG

1. הוספת Rest Assured לקובץ pom

```
<dependency>
   <groupId>io.rest-assured</groupId>
   <artifactId>rest-assured</artifactId>
     <version>5.1.1</version>
</dependency>
```

2. Best Test

נבנה את ה-class הזה כאשר כל הטסטים שלנו יירשו (אותה השיטה כמו בסלניום)

```
public class BaseTestAPI {

        @BeforeClass
        protected void setup() {
            RestAssured.baseURI = "<yourServer>";
            RestAssured.useRelaxedHTTPSValidation();
            RestAssured.requestSpecification = new RequestSpecBuil
                    .addHeader("Content-Type","application/jason")
                    .build();
        }
}
```

3. API Function class

נממש את כל הקריאות העיקריות POST, GET, DELETE, PUT ב-class הנקרא API functions שייתן לנו שימוש של כל הפרויקט שלנו.

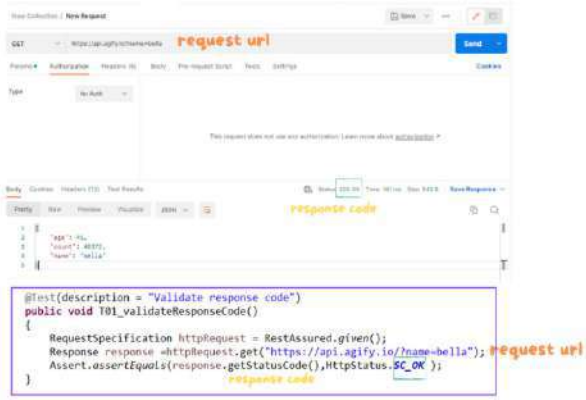
```
ic static Response Delete(String URL)
   {
        RequestSpecification httpRequest =
        RestAssured.given().header(Utils.readPropStartAPI("userName"
        Utils.readPropStartAPI("password"));

         Response response =  (Response) httpRequest
                    .header("Content-Type","application/json")
                    .when()
                    .delete(URL)
                    .then()
                    .extract()
                    .response();
          return response;
   }

ic static Response Get(String APIRequestName)
   {
        RequestSpecification httpRequest =
        RestAssured.given().header(Utils.readPropStartAPI("userName"
        Utils.readPropStartAPI("password"));
        Response response =   httpRequest
                    .header("Content-Type","application/j
                          .when()
                          .get(APIRequestName)
                          .then()
                          .statusCode(HttpStatus.SC_OK)
                          .contentType(ContentType.JSON)
                          .extract()
                          .response();

        return response;
   }
```

```
public static Response Post(String APIRequestURL, String body)
        {
                RequestSpecification httpRequest =
                RestAssured.given().header(Utils.readPropStartAPI("userName"),
                Utils.readPropStartAPI("password"));
                Response response =   httpRequest
                                .header("Content-Type","application/json")
                                .body(body)
                                .post(APIRequestURL)
                                .then()
                                .statusCode(HttpStatus.SC_CREATED).extract().response();
                return response;
        }
public static Response Put(String url, String body)
        {
                RequestSpecification httpRequest =
                RestAssured.given().header(Utils.readPropStartAPI("userName"),
                Utils.readPropStartAPI("password"));
                 Response response =  httpRequest
                                .header("Content-Type","application/json")
                                .body(body)
                                .when()
                                .put(url)
                                .then()
                                .statusCode(HttpStatus.SC_OK).extract().response();
                return response;
        }
```

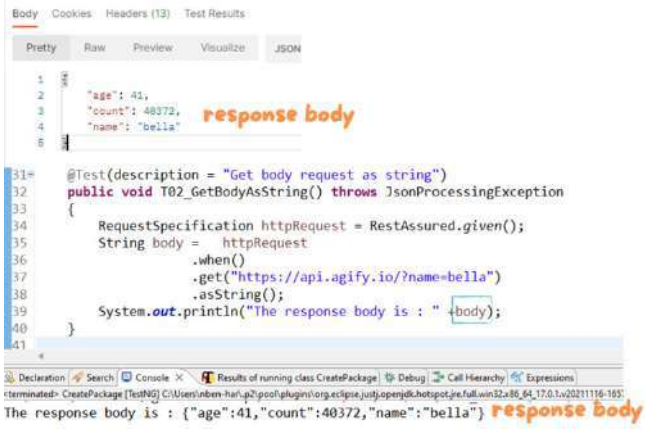
4. **בניה של טסטים (ארחיב בהמשך)**

מניפולציות על התשובה המתקבלת מהשרת

1. **בדיקת הסטטוס המוחזר ע"י הקריאה**
   בעזרת הקוד נוכל להשוות את הסטטוס המוחזר ע"י קריאת הבקשה



2. **קבלת התשובה כולה כמחרוזת**
   בעצם כאן אנחנו משתמשים ב-asString לאחר קריאת ה-get והשמה למשתנה מסוג String



3. **jsonPath() - חילוץ חלק מתוך התשובה המוחזרת מהשרת:**

ניתן להגיע לחלק מן התשובה, לא משנה כמה קובץ ה-Json המתקבל הוא מורכב, וניתן להשוות, להדפיס או ליישם למשתנים שנרצה את החלק המוחזר.


```
@Test(description = "Get age from response")
public void T03_GetAgeFromResponse() throws JsonProcessingException
{
    RequestSpecification httpRequest = RestAssured.given();
    Response response =httpRequest.get("https://api.agify.io/?name=bella");
    Assert.assertEquals(response.jsonPath().getString("age"),"41");
}
```

דרך שימוש שנייה ב-Json in "response.path()<"element to find"> בעזרת extract()

```
@Test(description = "Get age from response")
public void T03_GetAgeFromResponse() throws JsonProcessingException
{
    RequestSpecification httpRequest = RestAssured.given();
    Response response = httpRequest
                .when()
                .get("https://api.agify.io/?name=bella")
                .then()
                .extract().response();
        Assert.assertEquals(response.path("age"),"41" );
}
```

פלטור מתקדם – Gpath

Rest assured משתמש ב-GPath שזוהי שפה לאיתור אלמנטים בתוך Json.

בהנחה שזה קובץ ה-Json שאנחנו מקבלים כתשובה ונרצה לחפש בתוכו בצורה מתקדמת

שליחת בקשה לשרת – מספר דרכים

1. אפשרות ראשונה :
   שליחת גוף הבקשה כמחרוזת, שימוש ב-POST עם שליחת ה-URL המתאים.

בדיקות API אוטומטיות
עם אינטגרציה לסלניום | נתי צדוק

```
@Test(description = "create new user")
public void T01_CreateNewUser() throws JsonProcessingException
{
    OkHttpClient client = new OkHttpClient().newBuilder()
            .build();
    MediaType mediaType = MediaType.parse("application/json");
    RequestBody body = RequestBody.create(mediaType, "{\r\n    \"name\": \"Nati\",\r\n    "
            + "\"lastname\": \"Zadok\",\r\n    "
            + "\"description\": \"\",\r\n    \"address\": \"xxx\",\r\n    "
            + "\"city\": \"xxx\",\r\n    \"contry\": \"Israel\"\r\n}");
    Request request = new Request.Builder()
            .url("https://<yourserver>:<port>/api/user/add")
            .method("POST", body)
            .addHeader("Content-Type", "application/json")
            .addHeader("userName", "password")
            .build();
}
```

1
send the body as string
Using post method

2. שליחת גוף הבקשה כמחרוזת, שימוש ב-POST שבנינו ב-API Functions (אם עקבתם כמו שצריך 😊)

```
@Test(description = "Create new user")
public void T01_CreateNewUser() throws JsonProcessingException
{
    String body = "{\r\n    \"name\": \"Nati\",\r\n    "
            + "\"lastname\": \"Zadok\",\r\n    "
            + "\"description\": \"\",\r\n    "
            + "\"address\": \"xxx\",\r\n    "
            + "\"city\": \"xxx\",\r\n    "
            + "\"contry\": \"Israel\"\r\n}";
    Response response = APIFunctions.Post("https://<yourserver>:<port>/api/user/add", body);

    Assert.assertEquals(response.getStatusCode(),HttpStatus.SC_CREATED );
}
```

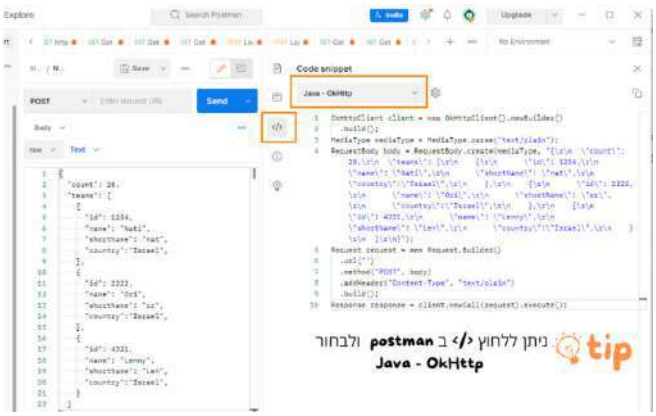
2
send the body as string
Using post method from API FUNCTION

3. שליחת אובייקט כבקשה ל-API

דרך זו אנו בעצם שולחים את הבקשה כאובייקט java מה שאנחנו בעצם שולחים זה "מראה" של קובץ Json לתוך class שניצור, דרך זאת היא אומנם מתוחזקת קוד יותר ודורשת מאיתנו מעקב על קבצי ה-json כלומר אם משתנה שם של ערך מסוים אנחנו בעצם צריכים לשנות גם את הקוד אצלנו. היתרון של זה שאנחנו לא צריכים להתעסק עם קבצי json או לפרסס אותם אנחנו פשוט עובדים pure java.

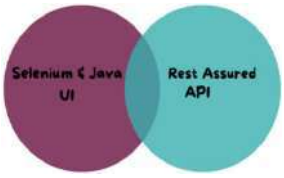
```
public class User {
    private String name;
    private String lastname;
    private String description;
    private String address;
    private String scripts;
    private String contry;
```

טיפ למשתמשי Postman



אינטגרציה עם סלניום

אחד הדברים שעזרו לי בבחירת Rest Assured על פני הכלים האחרים זאת אוטומציית ה-UI הממומשת אצלנו בקוד המבוסס Selenium & Java. כשהתחלתי לחשוב על מימוש לטסטים של ה-API, ראיתי במחקר שלי המון כלים אשר חלקם פשוטים יותר וחלקם טובים יותר, אך כשהבנתי שאפשר לעשות התממשקות מלאה בין האוטומציה שכבר קיימת לי בסלניום בחרתי את הכלי הזה.



ולמה?

- האפשרות להעמיק טסטים של GUI בבדיקות API על אותו המוצר (כבר נראה דוגמא)
- אפשרות ליצור טסט שמאפשר חיבור בין PI לבין UI (לדוגמא: יצירה של 100 משתמשים דרך קריאה של API ולאחר מכן לפתוח את הדפדפן ולהמשיך בטסטים של GUI על אותם משתמשים ואותה DATA
- אותה תשתית יכולה לשמש ל-2 הפרויקטים באוטומציה שניהם משתמשים Java, Maven & testNG

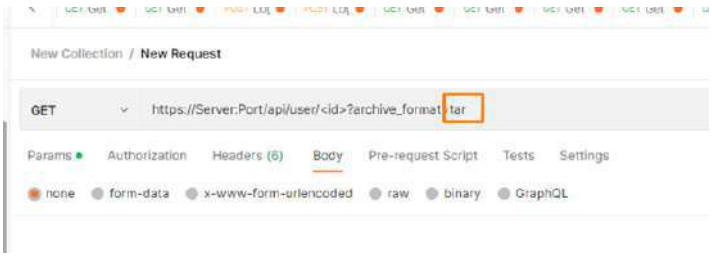
מבניות הפרויקט

המבניות שכבר הייתה קיימת לי עבור פרויקט הסלניום, עשתה את הדרך לבנות את הפרויקט API לקלה יותר. כל מה שהיה עלי לעשות זה להוסיף package חדש עבור ה-API, ולדברים משותפים כמו התממשקות עם דוחות אלור (Allure), קבצי מידע משותפים. כל ניהול קבצי XML המכילים את הטסטים, מנוהלים במקום אחד, ומכילים גם קובץ שישמש אותנו לראייה ב-Hi-level של כל הפרויקט הקיים שלנו גם עבור API וגם עבור UI.

האפשרות להעמיק טסטים של GUI בבדיקות API על אותו המוצר

בדיקות API

נוכל "לשחק" עם מנגנון ההורדה בתוכנה שלנו, לשלוח לדוגמא פורמטים שלא קיימים

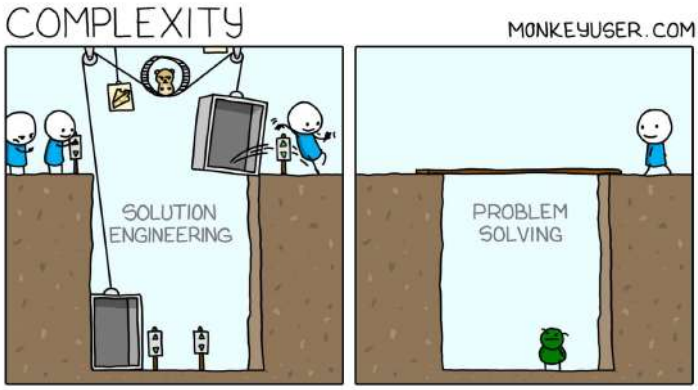



```
public Response Download(String packageID, String format) {
    {
        String downloadHeader =Utils.readPropStart("DownloadPackage")+packageID+"?archive_format="+format;
        System.out.println("url to download = " +downloadHeader );
        System.out.println("id to download = " +packageID );

        Response response = APIGet(downloadHeader);
        return response;
    }
}
```

בדיקות API אוטומטיות
עם אינטגרציה לסלניום | נתי צדוק

בדיקות UI

יש לנו את האפשרות ללחוץ על כפתורי ההורדה רק לאלו שאנחנו רואים



לסיכום

לא לוותר על בדיקות API, להבין שהדרך לכתוב תשתית אוטומטית לבדיקות אלה היא קלה ולא מסובכת בייחוד אם אנחנו כבר מתחזקים תשתית עבור בדיקות UI. וגם אם לא, המאמר מכיל רק חלק קטן ממה שאפשר לעשות עם Rest assured שהופך את הבדיקות ותחזוק הקוד לממש פשוט.

אני מאמינה שצוות QA טוב הוא זה שיש בו לפחות איש אחד שעיקר תפקידו הוא פיתוח תשתית הבדיקה, תמיד תנסו לחשוב מעבר למה שאפשר לפתח בעצמנו כי באמת הדברים לא מסובכים.

לכל שאלה ניתן לפנות אלי **בלינקדאין**

מה הסקרים אומרים?...

ניצן גולדנברג

מזה 7 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

CURRENT TRENDS IN QUALITY ENGINEERING & TESTING

טור זה מציג ממצאי סקרים המציגים מגמות מרחבי העולם לעיונכם.
הגליון הזה אנו מציגים חלק מתוצאות

World Quality Report 14th Edition | 2022-23

Fig 01 How much has each of the following areas improved since your organization adopted Agile/DevOps?

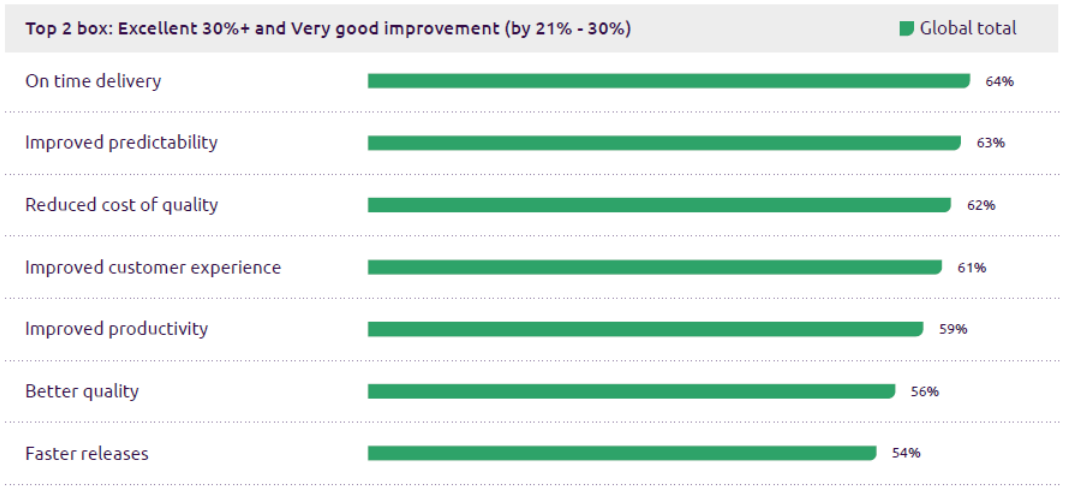


Fig 03 How important are the following QA skills when executing a successful Agile development program?

מה הסקרים אומרים?...

QUALITY AUTOMATION

Fig 05 What are the top three most important factors in determining your test automation approach?

Fig 06 What proportion (if any) of your team currently achieves the following benefits from test automation?

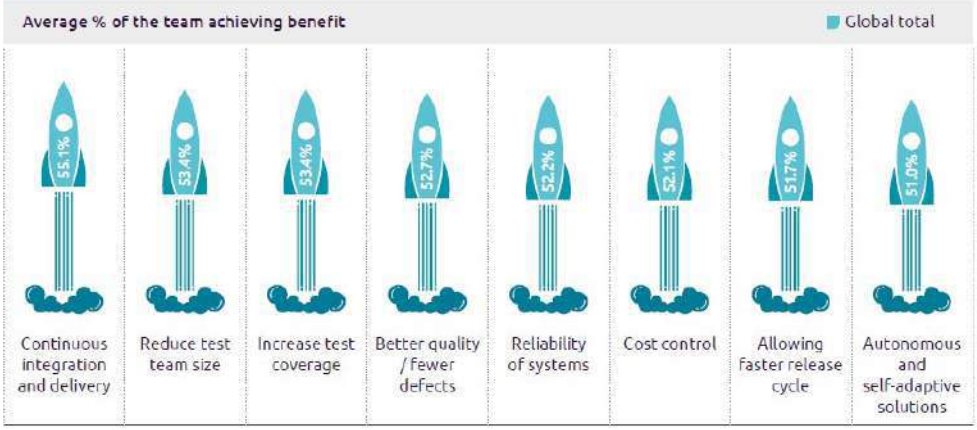
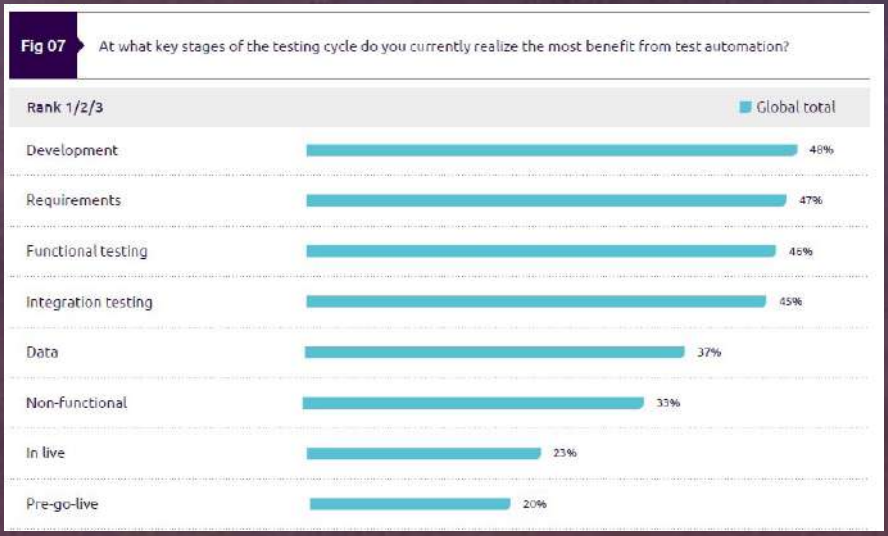


Fig 07 At what key stages of the testing cycle do you currently realize the most benefit from test automation?



מה הסקרים אומרים?...

INTERVIEWS BY SECTORS

- 7% Automotive
- 7% Manufacturing
- 6% Transportation
- 10% Consumer goods and retail/ Distribution and Logistics
- 18% High Tech including start ups, hardware vendors + Aerospace and Defense
- 11% Telecommunications, Media and Entertainment
- 15% Financial Services industry, including Capital Markets, Banking and Insurance
- 8% Healthcare and Life Sciences
- 11% Public Sector/ Government
- 7% Energy, Utilities, and Chemicals

INTERVIEWS BY JOB TITLE

CIOs	24%
QA Testing Manager	18%
IT Directors	17%
VP Applications	15%
CTO / Product Head	10%
VP / Director of R&D	10%
CMO / CDO	6%

עורכי הגיליון

בעל ניסיון של יותר מ-20 שנים כבודק ומנהל בדיקות במגוון חברות וטכנולוגיות במודלי פיתוח שונים. כיום טל יועץ ומדריך בדיקות עצמאי.

טל חבר ב-ITCB® וגזבר הארגון העולמי ISTQB®.

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

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.

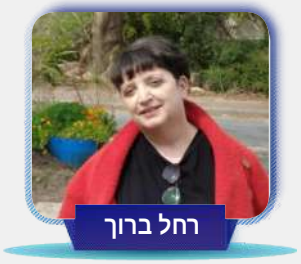
מובילת תחום, מרצה ומפתחת קורסי אוטומציה.

בשבילה החיים זה לא מספיק.

בעל 12 שנות ניסיון בפיתוח אוטומציות ובדיקות.

עובד כיום ב-Qwilt.

בודק תוכנה ב-Deep Instinct.



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

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

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.

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.

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.

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