מגזין
עולם הבדיקות
רבעון ראשון 2024 גיליון מס' 36
www.testingworld.co.il

תפקיד ה-Domain Expert בעולם האג'ייל
רון עובדיה

מה הסקרים אומרים לשנת 2023
ניצן גולדנברג

מהפרונט לבק של עולם הבנקאות
אילנה רז

אל תפחדו מקוד ספגטי בתוך פרויקט האוטומציה
אלכס קומנוב

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 36 של מגזין "עולם הבדיקות".

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

"תפקיד ה-Domain Expert בעולם האג'ייל" – רון עובדיה

אילנה רז מספרת לנו על השינוי שהיא עברה מעולם הבנקאות כנותנת שירות לבודקת תוכנה "מהפרונט לבק של עולם הבנקאות"

אלכס קומנוב מביא לנו עוד מאמר מעולם האוטומציה והפעם "אל תפחדו מקוד ספגטי בתוך פרויקט האוטומציה"

יש לנו תשבץ חדש ומאתגר בשבילכם בגליון הזה

בנוסף, יש לנו את הטורים הקבועים: מחפש צרות, בחן את עצמך, ראיון עם מנהל/ת בדיקות, האנציקלופדיה לבדיקות, עושים QA לקריירה, משולחנו של שביט, מה הסקרים אומרים על 2023 ונגיסה מ-TestIL.

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

magazine@testingworld.co.il

קריאה מהנה,
ניצן גולדנברג

עדכון מוסמכים בישראל

סה"כ מוסמכים בבורד הישראלי ITCB® – 10,051

מתוכם 878 "מצטיינים" אשר ענו מעל 90% תשובות נכונות.

174 מצטיינים ברבעון השלישי של שנת 2023

Visit our new website

WWW.ITCB.ORG.IL

תוכן העניינים

דבר העורך 2

האנציקלופדיה לבדיקות - Shift Left בבדיקות תוכנה
קובי יונסי 4

בחן את עצמך - שאלה | ירון צוברי 6

ראיון עם מנהל בדיקות מטיפיק | אביתר הכהן 7

בחן את עצמך - תשובה | ירון צוברי 8

תפקיד ה-Domain Expert בעולם האג'ייל | רון עובדיה 9

מחפש צרות - מסעותיי עם הדרישות | מיכאל שטאל 11

משולחנו של שביט - הניתוח הצליח, החולה מת
שביט ג'רסי 13

מהפרונט לבק של עולם הבנקאות | אילנה רז 14

עושים QA לקריירה.
מה כדאי (ומה לא כדאי), בכניסה לתפקיד חדש
איילת מלמד 16

אל תפחדו מקוד ספגטי בתוך פרויקט אוטומציה
אלכס קומנוב 18

מה הסקרים אומרים?... | ניצן גולדנברג 21

נגיסה מ-TestIL מפגשים לבודקי תוכנה | ניצן גולדנברג 23

בודקים בכיף 24

דף העורכים 25

עולם הבדיקות נכתב ע"י בודקים עבור בודקים

ITCB® מקדמים את קהילת הבודקים בישראל

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

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**.

Shift Left בבדיקות תוכנה

מידי פעם אני מוצא בודקי תוכנה ששואלים אותי לגבי המושג Shift Left שעולה מעת לעת בשיחות, בישיבות צוות או באופן מקרי כשמדברים על מיקומו של הבודק בשלב הפיתוח. הפעם החלטתי להאיר את המבט על מה זה אומר Shift Left ואיך זה קשור לאחד משבעת עקרונות הבדיקות.

קובי יונסי

בעל תואר ראשון מאונ' בר אילן בתחום הלוגיסטיקה והכלכלה.
מרצה מוביל במכללות הגדולות בישראל וראש החוג בטכניון בתעשייה מרצה הבית של חברות מיקור החוץ הגדולות במשק.
מייסד ומקים את המרכז המוביל לבדיקות תוכנה.
מלמד אנשים כיצד לחשוב יצירתי ולשפר את מיומנויות הבדיקה.

המושג Shift Left מתייחס לגישה בתחום בדיקות התוכנה שמצביעה על תזוזה של תהליכי הבדיקות מהשלב המאוחר שמגיע אחרי סיום הפיתוח אל שלבים מקדימים יותר בתהליך כמו שלבי האפיון והגדרת המוצר. המטרה העיקרית היא לזהות ולתקן את הבעיות של המוצר כמה שיותר מוקדם בתהליך פיתוח התוכנה ובדיקתה בכדי לשפר את איכות התוכנה. גישה זאת מתכתבת להפליא עם אחד העקרונות החשובים בבדיקות עליהם גם הרחבנו בעבר במגזין זה.

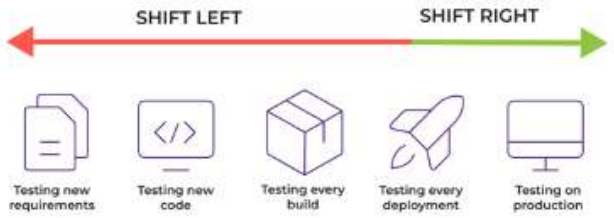
העיקרון נקרא בשם: Early Testing והפרשנות בעברית תהיה: **בדיקות מוקדמות חוסכות זמן וכסף**. משמעותו של כלל זה, בדיקות מוקדמות עשויות לזהות בעיות עוד בשלבים הראשונים של הפיתוח, כאשר עשוי להיות יחסית פשוט וזול לתקן אותם. כאשר נמצאים פגמים בשלבי פיתוח מוקדמים, ניתן למנוע התפשטותם ולחסוך בכך מאמץ רב בשלבים מאוחרים.

העיקרון הזה מדגיש את החשיבות של הרצת בדיקות ובדיקות יחידה מוקדמות, שימוש בכלים אוטומטיים לבדיקה וסימולציה, כדי לזהות ולתקן פגמים בשלבים הראשוניים יותר של הפיתוח. זה מפחית את הצורך בתיקון בגרסאות המתקדמות של התוכנה ומקנה גם יכולת לזהות ולתקן פגמים לפני שהם מצטברים וגורמים לבעיות גדולות יותר.

בעבר היינו רגילים לכך ששלבי הפיתוח בוצעו בצד שמאל של מודל V והבדיקות חיכו בסבלנות בצידו הימני.

עם השנים צוותי תוכנה הגיעו למסקנה שבאופן זה קשה מאוד להתמודד עם ציפיות ודרישות משתנות, אלו מובילים לחוסר תקשורת, הבנה שגויה ומשם לתקלות בלתי צפויות.

המשמעות העסקית היא שהוצאות הטיפול והמניעה של תקלות הן גבוהות מה שמייקר את עלות התוכנה ומאריך את זמן היציאה לשוק שחשוב לבעלי העניין (Time to the market)



גישת Shift Left מבליטה בין היתר את חשיבותו של בודק התוכנה במתן פידבק והעלאת חוסרים או טעויות במסמכי הדרישות.

פעילות חשובה שמסוגלת למנוע כניסת תקלות לתוך הקוד וגם מביאה לידי ביטוי את הניסיון של בודק התוכנה בהסתכלות על צרכי הלקוח וההתאמה לדרישותיו.

העקרונות המרכזיים של גישת Shift Left:

1. תודעה מוקדמת לבדיקה:

עיקרון זה מדגיש את הצורך להעביר את פעולות הבדיקה מהשלב המאוחר בתהליך הפיתוח אל השלב המוקדם יותר. צוותי הבדיקה צריכים להשתלב בפעילות בדיקות אשר תתחיל שלבים מוקדמים בתהליך הפיתוח ולבדוק כבר בשלב המוקדם ביותר ככל שמתאפשר.

2. אוטומציה של הבדיקות:

עיקרון זה דורש שימוש בכלים אוטומטיים לביצוע בדיקות. בדיקות אוטומטיות יכולות להיות יעילות ולהבטיח כיסוי נרחב של הקוד בזמן קצר.

3. שילוב של הבדיקות בתהליך הפיתוח:

הבדיקות צריכות להיות חלק מתהליך הפיתוח. זה כולל כתיבת בדיקות לפני כתיבת הקוד. בדיקות יחידה מתחילת התהליך ובדיקות אינטגרציה שמשתלבות בשלבים מוקדמים.

4. מעורר השקעה בצוותי הבדיקות:

עיקרון זה דורש הקפדה על העסקת צוותי הבדיקה בשלב מוקדם יותר בתהליך הפיתוח. זה עשוי לכלול הכשרה והקמת צוותי בדיקה מקצועיים שיוכלו להשתלב מיד בתחילת התהליך.

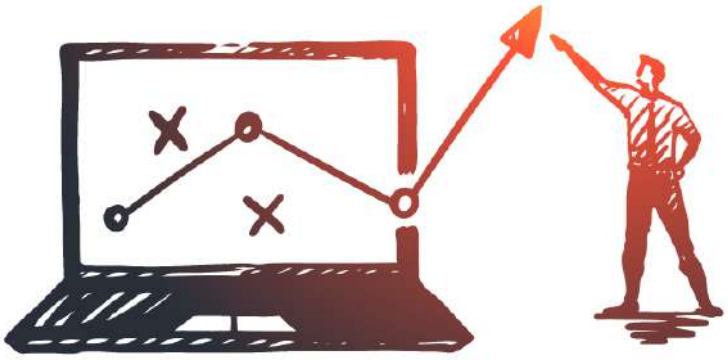
יתרונות של גישת Shift Left:

כלל ידוע הוא שתקלה שנמצאת לאחר עלייה לאוויר בזמן שהמוצר בקו הייצור תעלה ביחס של פי 100 מתקלה שזוהתה ונפתרה כבר בזמן כתיבת הדרישות.

מפה אנו למדים שיתרון מובהק של השיטה הוא צמצום עלויות של תיקון פגמים ושיפור משמעותי של תהליכי הפיתוח.

יתרונות נוספים שיש לגישה:

- זיהוי ותיקון של פגמים בצורה מהירה ויעילה לפני שעוברים לבדיקות דינמיות
- הגדלת אפקטיביות תהליכי הפיתוח והפחת זמן הפיתוח



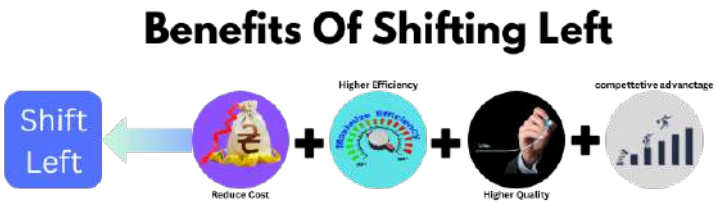
- הפחתת עלויות וזמני הבדיקות
- שיפור התקשורת בין צוותי הפיתוח לבדיקות
- זיהוי של פגמים שלא ניתן לאתר באמצעות בדיקות דינמיות כמו למשל: קוד מת, זליגת זיכרון, קוד לא יעיל (קוד ספגטי) וכו'.

חשוב לציין שלצד היתרונות ובכדי לשמור על תמונה מאוזנת קיימים גם חסרונות לגישה זאת

חסרונות של גישת Shift Left:

ישנן תקלות שלא ניתן לאתר מבלי הניסיון של לקוחות הקצה כמו בדיקות AB ובדיקות של שימושיות מול קהל רחב של משתמשים.

- עלות גבוהה של הטמעת הגישה בשלב ההתחלה מאחר ויש צורך בשדרוגים טכנולוגיים, בפרויקטים שכבר קיימים זה אף יהיה יקר ומורכב יותר.
- העברת הבדיקות שמאלה דורשת התאמה והקפדה על תהליכי עבודה חדשים ולכן ייתכן שנגלה התנגדויות מצד צוות הפיתוח המקורי.



כיצד נטמיע בחברה גישת Shift Left?

גופים רבים מתקשים לעשות את ההסטה שמאלה באופן מוצלח, כדי שזה יקרה מומלץ לחשוב על הנקודות הבאות שיכולות לאפשר את ההטמעה באופן מוצלח:

1. צרו מדדי איכות שניתנים לכימות

כדוגמת הגדרות מדויקות של תנאי כניסה מבוססים למשל על הצלחה של מבחני שפיות, כך שללא התקיימות המדד לא ניתן יהיה לקבל את הגרסה לבדיקה.

2. בדיקות מונחות סיכון

מאחר ולא ניתן לבדוק את כל המקרים בזמן קצר יהיה נכון לתעדף את מקרי הבדיקה החשובים יותר שימנעו רמת סיכון גבוהה מהמוצר.

3. שילוב אוטומציה ברמות הבדיקה הראשוניות

במרבית המקומות האוטומציה מתייחסת לממשק המשתמש ביצירת תסריטי End to End שמצריכים תחזוקה של צוותי האוטומציה.

בדיקות אלו נעשים בדרך כלל בשלב של בדיקות המערכת לפני שהיא נמסרת ללקוח כדי שיבצע את בדיקות הקבלה מצידו (UAT)

הגישה של Shift Left מעודדת השקעה בתהליכי אוטומציה ברמות הבדיקה הראשוניות בכדי להגיע לפחות בדיקות ידניות ברמות הגבוהות.

4. שינוי בתפיסה מאבטחת איכות למהנדס איכות

תפיסת הבודק בארגון חייבת להשתנות מאחד שצד באגים למי שתומך באיכות המוצר. הבודק החדש צריך להבין בפיתוח מונחה התנהגות (**B**ehavior **D**riven **D**evelopment) כדי להבין את התנהגות הלקוח ומעבר לכך להבין צדדים עסקיים של המוצר, היבטים טכניים בפיתוח ותחומים נוספים שיידרש אליהם כמו למשל הבנה בחוויית משתמש.

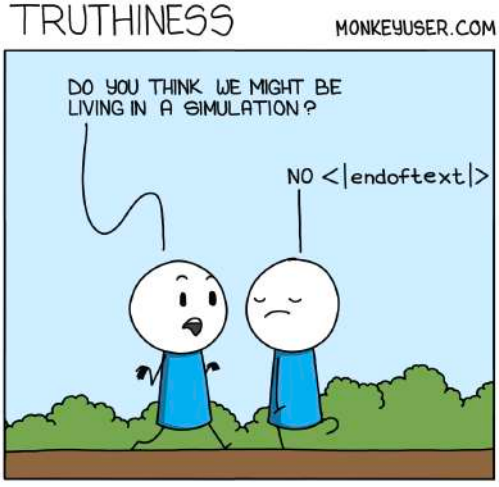
5. תמיכת ההנהלה

כמו כל שינוי בחברה, גם כאן אנחנו משנים את תפיסת האיכות בחברה, מלהטיל את כל האחריות על צוותי הבדיקות שפוגשים את המוצר הסופי בשלב מאוחר יחסית של הפיתוח (SDLC).

בגישה זאת האחריות נופלת על כל הצוות, שיתוף הפעולה הכרחי, זה כולל תהליכי הדרכה והטמעת הרעיון שלא רק אנשי הבדיקות לוקחים חלק באחריות על האיכות של המוצר הסופי. ללא תמיכה של המנהלים יהיה קשה עד בלתי אפשרי להטמיע רעיון זה אצל חברי הצוות ובעיקר אצל המתכנתים.

לסיכום,

כאשר אתם עוברים שמאלה על ידי מינוף טכנולוגיות בדיקת תוכנה מודרניות, אתם יכולים להשיג תוכנה בטוחה, אמינה ומאובטחת. על ידי העברת בדיקה שמאלה, אתם יכולים להפחית את עלות הבדיקה על ידי מציאת באגים מוקדם יותר, כאשר זה זול יותר, ובמקביל גם להפחית את מספר הבאגים שהכנסת לקוד מלכתחילה.
ולכן המסקנה של המאמר תהיה : **לזוז שמאלה – זה כדאי!**



בחן את עצמך | ירון צוברי

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.

אחת הפעילויות המעניינות והחשובות של בודקי התוכנה אחרי שביצעו ניתוח (Analysis - אֲנָלִיזָה) למערכת/אפליקציה היא לכתוב בדיקות. כתיבת הבדיקות מסתמכת לרוב על דרישות שהן חלק מבסיס הבדיקות (test basis) – בין אם הן כתובות כדרישות סטנדרטיות או סיפורי משתמש (user story/ies) שכוללים תנאי קבלה (acceptance criteria); תוך כדי כך, כבודקים, אנחנו נעזרים בטכניקות לעיצוב בדיקות (test design techniques), על מנת לייעל את תהליך הבדיקה ולערוך אותה באופן הנדסי.

מחלקות שקילות" (equivalence partitioning), או בשמה האחר "קבוצות שקילות" (equivalence classes) הינה אחת מהטכניקות הללו. טכניקת מחלקות השקילות מחלקת נתונים לקבוצות (הידועות כמחיצות שקילות) על סמך הציפייה כי כל הרכיבים של קבוצה נתונה יעובדו באופן זהה ויציגו התנהגות זהה בתוצאה הסופית. התיאוריה מאחורי טכניקה זו היא שאם מקרה בדיקה בודק ערך אחד מתוך מחלקת השקילות ומזהה פגם, פגם זה צריך להיות מזוהה גם על ידי מקרי בדיקה הבודקים כל ערך אחר באותה המחלקה - כנ"ל אם מקרה הבדיקה עובד עבור ערך מסוים, המקרה יעבוד עבור כל הערכים במחלקה. לכן מספיקה בדיקה אחת לכל מחלקת שקילות.

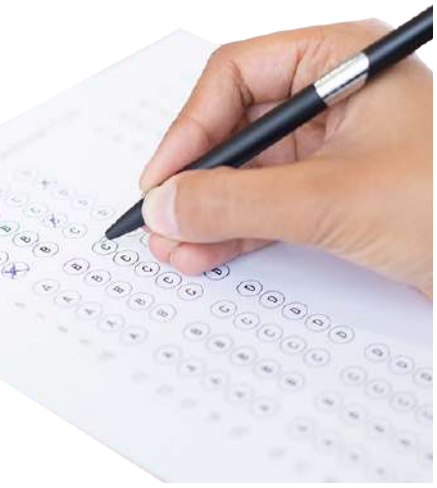
ניתן לזהות מחלקות שקילות עבור כל רכיב נתונים הקשור לאובייקט הבדיקה, כולל קלטים, פלטים, פריטי תצורה, ערכים פנימיים, ערכים הקשורים לזמן ופרמטרים של ממשק. מחלקות לא יכולות להיות חופפות, כלומר, כל ערך משתייך למחלקה אחת. כמו כן מחלקות לא יכולות להיות ריקות ולא יהיו ערכים שלא שייכים לאף מחלקה. מחלקה המכילה ערכים חוקיים נקראת מחלקה חוקית. מחלקה המכילה ערכים לא חוקיים נקראת מחלקה לא חוקית. ההגדרות של ערכים חוקיים ובלתי חוקיים עשויים להשתנות בין צוותים וארגונים.

על מנת להשיג אופטימיזציה של האפשרויות השונות הקיימות עבור מקרי הבדיקה ותנאי הבדיקה השונים – כיוון שאנחנו יכולים בנקל למצוא את עצמנו עם מאות או אלפי או אף מיליוני אופציות בדיקה שהן תוצר של השילוב של אותם פרמטרים, ערכים, קלטים, פלטים וכו' – נחלק אותם למחלקות השקילות ונכלול את אותם הערכים במחלקות/קבוצות אלו, מתוך מטרה (עבור האופטימיזציה) שבדיקת ערך אחד מכל קבוצה תספיק עבור הכיסוי של אותה הקבוצה. (למידע נוסף, מומלץ לקרוא את **פרק 4 תת-פרק 4.2.1** של תוכנית ההכשרה הבסיסית של אירגון ISTQB).

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

אתה בודק טופס חיפוש דירות פשוט הכולל שני קריטריוני חיפוש בלבד:

- קומה (עם שלוש אפשרויות אפשריות: קומת קרקע; קומה ראשונה; קומה שנייה ומעלה)
- סוג גינה (עם שלוש אפשרויות אפשריות: ללא גינה; גינה קטנה; גינה גדולה)



רק בדירות בקומת הקרקע עשויות להיות גינות. לטופס יש מנגנון אימות מובנה שלא יאפשר לך להשתמש בקריטריוני החיפוש שמפרים כלל זה.

לכל בדיקה יש שני ערכי קלט: קומה וסוג גינה. אתה רוצה להשתמש במחלקות שקילות (equivalence partitioning) בכדי לכסות כל קומה וכל סוג גינה בבדיקות שלך.

מהו המספר המינימלי של מקרי בדיקה הנדרש, בכדי להשיג 100% כיסוי באמצעות טכניקת מחלקות שקילות?

א. 3
ב. 4
ג. 5
ד. 6
בחר אפשרות אחת

*לצפייה בתשובה המפורטת - דפדפו לעמוד 8

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

ראיון עם מנהל בדיקות - אביתר הכהן מטיפיק

אביתר הכהן

קוראים לי אביתר הכהן בן 28, נשוי לטליה וגר בבאר יעקב, מרצה למקצוע הבדיקות ויזם בתחום הגיימינג. כיום אני משמש כ-Automation & QA Team Lead בקבוצת ה-Apps בחברת גיימינג חינוכי - "מטיפיק". אפשר לומר שזאת די סגירת מעגל בשבילי לעבוד במוצר חינוכי שנוגע לחינוך של הרבה ילדים בישראל וברחבי העולם כי בעברי הייתי מורה בבית ספר. המוצר של החברה בה אני עובד מבוסס יוניטי, מה שמעניק חווית משחק ייחודית ונעימה למשתמש. המשתמש משחק אווטאר המסתובב בין איים בעלי אפיון שונה, אי של הרפתקה שבה המשתמש מכיר דמויות ומפלצות חדשות, אי שבו המורה של אותה הכיתה של הילד או ההורה של המשתמש יכולים לשייך לו כשיעורי בית משחקונים שהם אתגרים מתמטיים, משחקונים שמספרים סיפור ומציגים בעיות מתמטיות למשתמש שאותם הוא צריך לבחור. התוכן מותאם ע"פ הסילבוס של משרד החינוך במדינות בהם אנחנו משווקים. אין ספק שזה מוצר שאביתר של גיל 9, היה רוצה לגדול לדור שיש בו את "מטיפיק". עובדה שחשוב לדעת עלי? אני גיק מושבע, לא מזמן סיימתי מחזור של קורס בדיקות תוכנה במכללה שבה אני מלמד ואחד הסטודנטים נתן לי קומיקס אספנים של באטמן מהאוסף שלו.

במה עוסקת הקבוצה וכיצד היא בנויה?

הקבוצה שלי עוסקת בבדיקת המוצר מהצד היותר גיימינג'י שלו, הפיתוח של הקבוצה שלנו הוא פיתוח ע"י UG Engine הווי אומר שאנשי ה-QA בקבוצה כולם גיימ-טסטרים.

המוצר הוא קרוס פלטפורם ולכן ההתמקצעות של כל אנשי הצוות זה מובייל, וב ואפילו דסקטופ.

הקבוצה משלבת בין בודקי אוטומציה לבודקים ידניים.

מהן הדרישות הניהוליות והעסקיות מאנשי הבדיקות וממך כמנהל?

כנות מקצועיות וחקירה עד לפרטים הקטנים, עבודה בשעות מגוונות מאוד, בגלל העובדה שאנחנו חברה גלובלית. אסביר איך כל הדרישות הללו נשזרות אחת בשנייה, כנות ומקצועיות זה חלק מהאיכות של הגרסה, רק ע"י כנות מלאה אוכל לסמוך על מילה של עובד בנוגע לפיצ'ר שאני פחות נגעתי בו כמנהל, חקירה? אם איש בדיקות או אשת בדיקות מוצאים תקלה אני מצפה שהם ייצרו איתי כאשר הם כבר חקרו ויש להם פרטים או אפילו תחושת בטן ויחד נחקור.

עבודה בשעות מגוונות - זה בעיקר נובע בגלל הפרשי זמנים בין הקבוצות.

איך נראה יום העבודה שלך?

אשתי תמיד אומרת פגישות, פגישות, פגישות... ואז שואלת מתי אני עובד ובתכלס זה די מצחיק בהתחשב בזה ש-90 אחוז מהזמן השבועי שלי אני מעביר בפגישות. אם באמת נכנס לזה, את הבוקר שלי אני משתדל לפתוח אחרי כארוחת בוקר קלה וכוס קפה עם המשפחה, תפילה טובה ופעילות.

לעבודה אני מגיע בערך בשעה 10:00, ודבר ראשון עובר על מיילים ודברים דחופים וישר פונה פגישות דיילי. לאחר שכל אחד מחברי הצוות מסונכרנים במשימות היומיות שלהם, אני מתחיל במשימות שלי בכך שאני עובר על דוחות האוטומציה מהרצות הלילה בחוות הבדיקות (AWS – Device farm) וסוקר את הדוחות ומתוך כך מעצב את תהליכי הבדיקה ובמידה ונמצא באג בסביבת הייצור (Production) הוא מקבל תיעדוף וזמן משלו בלו"ז היומי. כמו כן, בין השעות 14:00-15:00 אני חייב את הפסקת הקפה שלי וקריאה של מאמר מדעי בתחומי הטכנולוגיה וכמובן ממשיכים בשגרת היום.

כראש צוות, העבודה לרוב לא נגמרת ואני מוצא עצמי עם מחשב פתוח גם בשעות הערב.

באילו אתגרים נתקלים הבודקים שלך?

שאלה ממש טובה, רציתי להתייחס בהתחלה לסוגי בדיקות מאתגרות יותר הקיימות בעולם הגיימינג.

אבל אני חושב שהאתגר הכי גדול שלנו, אם נשים בצד את הדיון של סביבות עבודה שזה האתגר הגדול של כל מקום עבודה, לדעתי אתגר גדול מאוד זה תקשורת רחבה בין הרבה גורמים שמפתחים פיצ'רים ובודקים אותם. כי רוב פיצרים אצלנו מפותחים בין הרבה צוותים, פיצ'ר מתחיל אצל הפרודקט ואז עובר לגיימדיזיינר שם הפיצ'ר מתחלק בעבודה בין מספר קבוצות ולכל קבוצה יש אחריות על הפיצ'ר במלואו וגם על הצד שלהם שאותו הם מפתחים. מה שעושה את העבודה למורכבת מאוד.

באילו אתגרים ניהוליים הנך נתקלת?

בעיקר ניהול של צוות גלובאלי, לדעת ולהכיר סטטוס של כל משימה שהחברה שלי עובדים עליה על מנת לדעת אם להוריד משימות בתיעדוף לקראת גרסה או לקבל עדכונים קרוס אטלנטים בין מפתחים מקבוצות אחרות שמצריכות עבודה זהירה ונוספת של אנשי ה-QA.

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

אגע בשני נקודות, בדיקות UI בצורה מבוקרת ונכונה - לבדוק UI באתר או באפליקציה זה דבר אחד, לבדוק UI במשחק גיימינג זה מאוד מורכב יש המון אלמנטים על המסך וזה מצריך היכרות עמוקה על המוצר, היכרות על כל אנימציה שיש וכמובן אלמנטים משתנים מתוך הקושי הזה, אחד הדגשים בצוות זה שימוש של בדיקות חוקרות, מעבר לסימון ומעבר על תסריטים, לכל בודק בכל פיצ'ר ובכל גרסה יש זמן של בדיקות חוקרות שבהם הוא מחולק לאזורים ובודק את האזורים במעין חופשיות שתסריט כתוב לא מאפשר.

הדבר השני זה אוטומציה, בהתחלה כאשר בנינו את התשתית לפרויקט האוטומציה התחלנו לחקור על אוטומציה מול מנוע גיימינג יוניטי וגילינו שבארץ אין פיתוח אוטומציה על גבי משחק יוניטי.

התחלנו לחקור ולבנות מתודולוגיות עבודה על בסיס שרת מידע משרת הדיסקורד.

יש הרבה עבודת תיעוד שלנו בנושא הזה וחקירה שזה מצד אחד מאוד טוב ומלמד אבל מצד שני זה פחות קל.

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

דבר ראשון בעיני חשוב מאוד תמיד להגיד "תודה", "בבקשה" כל משימה שהצוות מקבל אני מעביר להם את הביקורת החיובית וגם השלילית. מעבר לזה אנחנו משתדלים אחת לגרסה להיפגש לפגישת רטרו לדבר על דברים שהפריעו ולשפר את התהליכים בנינו.

שיפור תהליכים קשור קשר הדוק ליוזמה של אנשי הצוות וככל שאנשי הצוות יזמו יותר הצעות ורעיונות לשפר את התהליך שלנו אני מתפרע איתם ומשיג להם את "הזמן" ומה שצריך בשביל שהקבוצה שלנו תתקדם. לא להסס לשאול שאלות, משלב הקורס עד ה-6 שנים הראשונות, כל שאלה שעולה לך לראש תשאל, פגישות טכניות? תשאר! אתה תלמד הרבה ותפתח שיח טוב עם המפתחים.

ראיון עם מנהל בדיקות -אביתר הכהן מטיפיק

חשוב מאוד להיות בקשר טוב עם מפתחים, זה יעזור הרבה בידע הטכנולוגי שלכם.

ממה היית ממליץ להימנע?

מאוכל וחטיפים בהייטקס, בקיץ זה בכלל מטורלל כל סוגי הגלידות האלה.

ואם נתייחס לדברים שקשורים לתעשייה, הייתי ממליץ לבודקים מתחילים להעמיק מאוד בלימודי טכניקות בדיקה וסוגי בדיקה לפני שרצים ללימודי אוטומציה.

מתכנת אוטומציה טוב, זה קודם כל איש QA פנטסטי

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

פספורט קבוצתי

סוג המוצר הנבדק: משחק מבוסס יוניטי קרוס פלטפורם שמלמד בעצמו ועוזר לילדים ללמוד מתמטיקה, המון חברים וחברות שלי בעולם ההוראה משבחים את היעילות של האפליקציה וכמובן מספרים לי על באגים ידועים.

גודל קבוצת הבדיקות: מגוון מאוד, כרגע 4 בודקים ו-2 מתכנתי אוטומציה.

וותק הבודקים: עד שלוש שנים

מבנה הקבוצה: קבוצת ה-QA נמצאת תחת קבוצת ה-Apps תחת ניהול צוות של ר"צ הקבוצה שלנו.

סוגי בדיקות: אנחנו מבצעים את כל סוגי הבדיקות מהקלאסיות יותר (פונקציונאליות, שימושיות, עומסים, אוטומציה, בדיקות מובייל, FPS, שימוש חומרה, בדיקות חופשיות) עד כאלה שפחות שמענו עליהם כמו בדיקות גורילה ומאנקי במבנה האוטומציה שלנו.

שיטת עבודה: אנו עובדים בשיטת אג'ייל.

כמות המוצרים הנבדקים וקצב שחרור גרסאות: מוצר אחד אשר נבדק על מגוון רחב של סביבות, שחרור גרסה אחת לחודש.

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 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-4.2.1 - "השתמש בקבוצת שקילות בכדי ליצור מקרי בדיקה"; 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 כתיבת בדיקות, חושף אותנו הבודקים להשתמש בטכניקות (בעלות אופי הנדסי ו/או מתמטי) לעיצוב בדיקות. טכניקת "קבוצות שקילות" הינה טכניקה מאד נפוצה, ולעיתים אף נשתמש בה יחד עם טכניקת נוספת: "ניתוח ערכי גבול" (boundary value analysis). היכולת להגדיר קבוצות של ערכים שקולים, בעלי אופי דומה, ובדיקת מקרה אחד מכל קבוצה, מאפשר לנו לצמצם את כמות מקרי הבדיקה האפשריים, מתוך הנחה שמספיק מקרה אחד מייצג לקבוצה שכן אם מקרה אחד יכשל, אזי יכשלו – מאותה הסיבה – כל אותם המקרים של אותה הקבוצה.

הבה נבחן את השאלה והתשובות:

בשאלה שלנו ישנם שני קריטריונים (סוג קומה, וסוג הגן), וכל קריטריון מכיל 3 אפשרויות. חישוב כמות מקרי הבדיקה האפשריים, הינו מכפלת האפשרויות (3X3 = 9).

מכיוון שנוסף הכלל ש"גינה קטנה" ו"גינה גדולה" יכולות ללכת רק עם "קומת קרקע", אנחנו צריכים לייצר שני מקרי בדיקה עם "קומת קרקע" המכסים את שתי המחלקות "סוג הגן". על-מנת להשיג 100% כיסוי באמצעות טכניקת מחלקות שקילות, אנו זקוקים לשני מקרי בדיקה נוספים, בכדי לכסות את שתי האפשרויות האחרות של "סוג הקומה", ואפשרות "סוג הגן" שנותרה - "ללא גינה".

אשר על כן, אנו זקוקים בסך הכל לארבעה מקרי בדיקה:

מקרה בדיקה 1: קומת קרקע, גינה קטנה

מקרה בדיקה 2: קומת קרקע, גינה גדולה

מקרה בדיקה 3: קומה ראשונה, ללא גינה

מקרה בדיקה 4: קומה שנייה ומעלה, ללא גינה

א. התשובה אינה נכונה.

ב. התשובה נכונה.

ג. התשובה אינה נכונה.

ד. התשובה אינה נכונה.

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

תפקיד ה-Domain Expert בעולם האג'ייל

רון עובדיה

בשנים האחרונות חלה התפתחות משמעותית מאד בתחום בדיקות התוכנה. במאמר זה אנסה להביא סקירה (ממקור ראשון!) שמציגה את תהליך ההתפתחות המרשים הזה (שקרה אצלנו בחברה, אצלי בצוות ולא רק על הנייר). נבחן ביחד כיצד השתנה תפקיד בודק התוכנה לאורך השנים, מביצוע בדיקות ידניות בסיסיות ופשוטות, ועד לתפקיד שהגענו אליו, Domain Expert (מומחה התחום) שהוא בעל ידע טכני ועסקי עמוק ורחב בתחום המוצר שעל איכותו הוא אחראי.

אני רון, מנהל צוותי QA כ-15 שנים בתחום ה-web. במהלך השנים הקמתי ופיתחתי לא מעט צוותים ואנשים במגוון תפקידים. החל מבודקים ידניים ועד לאנשי אוטומציה ו-DE (שארחיב על התפקיד בהמשך). אוהב שינויים ואתגרים ומנסה לעשות אותם תמיד בדרך של שיתוף פעולה.
אוהב כדורגל ואת המשפחה ובין לבין מבלה בהופעות של עידן עמדי 🙂.

רון עובדיה
נשוי ללידר ואבא ל-4 בנים.
אוהד ושחקן כדורגל מושבע (יאלה בית"ר), חובב טכנולוגיות בשאר הזמן.
מנהל את מחלקת ה-QE בחברת SeatGeek ומעל ל-15 שנים בתחום.
אוהב אתגרים, שינויים ומשתדל לראות את האור בקצה."

מבדיקות ידניות לאוטומציה

בשלבים הראשוניים של פיתוח התוכנה אצלנו, בודקי התוכנה התמקדו אך ורק בביצוע בדיקות ידניות פשוטות ובסיסיות כדי לאמת פיתוחים חדשים במוצר ולזהות באגים שפוגעים בתפקוד הנדרש מהמערכת. אולם, ככל שהתוכנה הפכה למורכבת, מתוחכמת ומסועפת יותר עם השנים ועם ההתקדמות הטכנולוגית, נוצר צורך הולך וגובר להרחיב ולשפר את היקף ואיכות בדיקות התוכנה.

לפני כעשור, עוד בזמן שעבדנו בשיטת RUP - Rational Unified Process ושיחררנו גרסה כל שלושה חודשים, בא אליי מנהל המוצר כמה שעות לפני שחרור הגרסה עם בקשה "קטנה" ורצון לתמוך בסוג נוסף של מטבע שלדבריו, לדברי מנהל הפיתוח ולדברי ה-CTO של החברה, אין כמעט סיכון, "תוך שעתיים אתם מסיימים את הבדיקות". רק לאחר 3 שבועות(!) מרגע זה שיחררנו את אותה גרסה לאחר אין ספור מהלכים של בדיקות רגרסיה מקיפות אחרי כל באג משמעותי שמצאנו ותיקנו. זאת רק דוגמא קטנה לצורך מאד גדול של בדיקות רגרסיה שליווה אותנו לאורך שנים.
כמות הבדיקות גדלה משמעותית וכך גם כמות העובדים בצוות. היינו מאד יצירתיים ומתוחכמים אבל תמיד הרגשנו כ"שומרי הסף", אלו שנמצאים בסוף התהליך ומחפשים איפה פספסו משהו. בשלב מסוים הבנו שללא אוטומציה יהיה לנו מאד קשה לשמור על איכות המערכת ונכנסו לעולם האוטומציה. לאחר השקעה של מספר חודשים שבה בנינו תשתית אוטומציה והכשרנו את כל אנשי ה-QA לתפקיד של מהנדסי אוטומציה, הצלחנו להגיע לכיסוי משמעותי של המערכת על ידי בדיקות אוטומטיות שעזרו לנו להוציא גרסאות איכותיות יותר ובקצב גבוה בהרבה. ההתרגשות הייתה גדולה מאד!, במהלך הדרך למדנו והשתכללנו (השתדרגנו מסלניום ל-Playwright), עברנו מ-RUP ל-Agile scrum ואנשי ה-QA היו חלק אינטגרלי מצוותי הפיתוח, אבל עדיין משהו היה חסר.

> "תמיד הרגשנו כ"שומרי הסף", אלו שנמצאים בסוף התהליך ומחפשים איפה פספסו משהו"

שמעתם על Shift-Left?

אחרי שייצבנו את השינויים הרבים ואת התהליכים החדשים, נוצרה הבנה חדשה לגבי הצורך לשלב את אנשי ה-QA הרבה יותר מוקדם בתהליך פיתוח המוצרים השונים, ולא רק בשלבים המאוחרים והסופיים של סיום הפיתוח. אנשי ה-QA כבר היו חלק בלתי נפרד מצוותי הפיתוח והוחלט לשלבם כבר בשלבים הראשונים של העבודה המתבצעת בשולחנות מנהלי המוצר.

על המושג Shift Left שמעתי לראשונה בכנס ה-QA GEEK WEEK של ג'ון ברייס וזה סקרן אותי מאד והתחבר לשינויים שהלכנו לקראתם.

במקום ההתמקדות המסורתית בביצוע בדיקות בסוף תהליך הפיתוח, החל המיקוד לעבור בהדרגה לשלבים המוקדמים של תכנון הפיצ'ר, תכנון תכנית הבדיקות והקריטריונים לשחרור של כל פיצ'ר ופיצ'ר. כך נוצר לראשונה מודל של אחריות משותפת והדדית לאיכות המוצר בין כלל האנשים בצוות הפיתוח ולא רק של אנשי ה-QA עצמם.
המפתחים היו שותפים מלאים לבדיקות שנעשו במהלך הפיתוח ואנשי ה-QA שיתפו והשפיעו מניסיונם לאורך כל הדרך ולא רק בסיום כשהפיצ'ר כבר גמור. עם זאת, עדיין היו קשיים רבים בשמירה על ראייה כוללת של המוצר וקבלת אחריות כוללת לכל פיצ'ר של צוותי הפיתוח.

עלייתו של ה-Domain Expert!

כדי לנסות ולפתור את הקושי בשמירה על ראייה רחבה של המוצר ואיכותו וכן לקיחת האחריות המלאה של צוותי הפיתוח, החל אט אט להתפתח תפקיד חדש שכינינו "Domain Expert" בעולם של בדיקות/איכות התוכנה. Domain Expert, או בקיצור QADE הוא בודק תוכנה שמתמחה באופן ספציפי ומעמיק באזור מסוים של המוצר ויכול לספק מידע רב וכן הכוונה אסטרטגית כוללת ורוחבית בין כל הצוותים השונים שעובדים יחדיו על אותו מוצר. הוא זה שיזהה את הסיכונים בזמן הפיתוח ויהיה שותף פעיל במטרה לשמור ולשפר את איכות המוצר.

כמות הבדיקות הידניות ירדה בצורה משמעותית ונעשתה ע"י כלל האנשים בצוות, כמו כן הבדיקות האוטומטיות נכתבות ע"י כולם. כבר בשלבי האפיון, מחליטים ביחד איזה בדיקות נדרשות עבור אותו הפיצ'ר וראש הצוות מחלק את משימות הבדיקה בין כולם.
הגדרת ה-Acceptance Criteria גם היא נעשית בשלב מוקדם ומוסכמת על כל המעורבים (QADE, המפתח, איש ה-Product וראש הצוות). ה-QADE אחראי על כתיבת תכנית הבדיקות (Test Plan) בשיתוף פעולה עם המפתח, ועליו לוודא שהתוכנית אכן בוצעה ואין שום מגבלות או באגים המונעים את שחרור הפיצ'ר בגרסה הקרובה. לאחר כשנה וחצי שהצוותים עבדו בצורה טובה ובשיתוף פעולה עדיין משהו היה חסר...

QADE ⟵ DE

לאחר שצוותי הפיתוח הבינו את חשיבות הבדיקות והאיכות, למדו לכתוב תכנית בדיקות ו-Acceptance Criteria, החלטנו לעבור לשלב הסופי (כרגע..) ולהכריז על התפקיד Domain Expert!
אנשי ה-DE הופרדו מהעבודה היומיומית עם המפתחים והחלו לעבוד ישירות מול אנשי ה-Product וראשי הצוותים. אין יותר "Gate Keepers"!

תפקיד ה-Domain Expert בעולם האג'ייל רון עובדיה

כל נושא ה-QA התחזק והוטמע בצורת העבודה של צוותי הפיתוח והאחריות עברה לחלוטין למפתחים.

המשימה שלנו כצוות Domain Expert היא להבטיח את ה-Delivery של המוצר שלנו באיכות גבוהה על ידי קירוב וגישור של הפער בין בעלי העניין במוצר לצוותי הפיתוח. נשיג זאת על ידי הבנה מעמיקה של התחום העסקי, אימוץ חשיבה מערכתית ומתן Acceptance Criteria ברורים ומוגדרים היטב. בנוסף, אנו נתרום לשיפור מהימנות המוצר וביצועיו באמצעות ניטור מקיף הן בתהליך הפיתוח והן בסביבת הלקוח וכן סימון אזורי סיכון על מנת להתמקד בהם בתהליך הפיתוח, הבדיקות והניטור.

ה-DE מביא איתו פרספקטיבה עסקית רחבה ומעמיקה בהרבה, בניגוד למיקוד הטכני יותר המאפיין את מהנדס האוטומציה או ה-QA המוכרים.

יתרונותיו של ה-DE

המודל שבנינו עבור ה-DE מביא עמו יתרונות רבים ומשמעותיים:

- מיקוד משופר ומעמיק בתחום המומחיות הספציפי שלו, התורם ערך רב יותר לעומת יכולות כלליות ורוחביות.
- גמישות גדולה וטובה יותר בפיתוח תכונות ויכולות חדשות למוצר, בזכות שילוב מוקדם ומעמיק יותר של בדיקות איכותיות.
- פרספקטיבה עסקית רחבה ומעמיקה יותר, העוזרת לזהות תלויות וסיכונים מוסתרים שלא היו ניתנים לאיתור בעבר.
- גישה פרואקטיבית לאיכות, המונעת בעיות ובאגים רבים עוד בטרם הם מוטמעים במוצר.
- שיתוף ידע נרחב ואימון שוטף של אנשי הפיתוח ע"י ה-DE, תורמים לשיפור משמעותי של היכולות המקצועיות בכל הארגון.
- מעורבות גדולה יותר בתהליכי הפיתוח, התורמת לשביעות רצון גבוהה יותר ולשימור טוב יותר של עובדים.
- DE הוא כיום גורם מפתח וקריטי להצלחת המוצר ואיכותו הסופית.

חסרונותיו של ה-DE (כמו בכל דבר, גם פה יש)

- עלויות גבוהות יותר הכרוכות בהכשרה ותחזוקה של DE ברמה גבוהה.
- תלות מוגברת במספר מצומצם של מומחים בודדים עלולה ליצור "צווארי בקבוק" בתהליכי הפיתוח.
- דרישה לרמת מומחיות גבוהה עשויה להקשות על גיוס ושימור של אנשי מקצוע איכותיים.
- עומס מידע ומורכבות גדולים יותר דורשים מעורבות צמודה יותר של הנהלת הצוות.

מסתכלים קדימה...

ההתפתחות הטבעית של תפקיד ה-DE הולכת לכיוון יותר אחראי/ניהולי ועם הזמן, עושה רושם שאנשי ה-DE יהוו חלק בלתי נפרד מהנהלת הצוותים בכל הקשור לאיכות המוצר. הטמעת תהליכים, כלים ושיטות עבודה להבטחת איכות איתנה ואמינה יהיו באחריותם הישירה.

בנוסף, אני צופה כי תפקיד ה-DE ימשיך להתרחב ולכלול היבטים נוספים מעבר לאיכות המוצר עצמו. לדוגמה, הבטחת חוויית משתמש אופטימלית ויעילה (UX), ניטור ביצועים ויציבות המערכת בסביבות הלקוח (Reliability & Performance) והיבטי אבטחת מידע ופרטיות (Privacy & Security).

כל אלו יאפשרו ל-DE לראות את התמונה הכוללת ולוודא שהמוצר עונה על מגוון רחב של דרישות, מעבר לתקינות המוצר, ובכך יתרום עוד יותר להצלחתו לטווח ארוך.

> "המסע שעברנו יחד בשנים האחרונות, מבדיקות תוכנה ידניות למודל מתקדם של שיתוף פעולה בין-תחומי ומומחיות עמוקה, היה אחד הדברים המלהיבים והמעצבים ביותר"

לסיכום

המעבר ההדרגתי מביצוע בדיקות תוכנה ידניות ובסיסיות בלבד, לשילוב משמעותי ומעמיק של DE בכל תהליכי הפיתוח, מהווה ללא ספק התפתחות חשובה ומשמעותית בתחום בדיקות התוכנה בשנים האחרונות. DE, הודות לחשיבה היצירתית וממוקדת האיכות, בעל הידע הרב, המעמיק והמקצועי שצבר ב-Domain הספציפי עליו הוא אחראי, תורם ערך עצום לשיפור האיכות הסופית והצלחתו העסקית של המוצר שלנו. ניתן בהחלט לצפות שתפקיד מרכזי וחשוב זה ימשיך להתבסס ולהוביל את המהפכה בתחום בדיקות התוכנה גם בעתיד.

המסע שעברנו יחד בשנים האחרונות, מבדיקות תוכנה ידניות למודל מתקדם של שיתוף פעולה בין-תחומי ומומחיות עמוקה, היה אחד הדברים המלהיבים והמעצבים ביותר עבורי בתפקיד.

אני זוכר את הלילות שבהן היינו נשארים עד השעות הקטנות במטרה לשחרר גרסה איכותית ככל האפשר. באמת שהיה כיף אבל אני בטוח שהצוות היה רוצה לנצל את השעות האלו בעיסוק בתחביבים אחרים. ואז, כשהתחלנו לשתף פעולה עם צוותי הפיתוח ולהטמיע את הבדיקות כבר בשלבי התכנון הראשוניים, פתאום הרגשנו שאנחנו שותפים אמיתיים לאיכות המוצר ויש לנו השפעה אמיתית! ידעתי שהקול שלנו נשמע והתרומה היא אמיתית חשובה.

כשגיבשנו יחד את המודל של ה-DE, פתאום הייתה לנו הזדמנות לא רק לבדוק, אלא לחקור לעומק ולהתמקצע כל אחד ב-Domain שלו, להבין את המטרות והאתגרים של המוצר ולקחת אחריות על הצלחתו.

התרבות החדשה שיצרנו - של שיתוף, למידה הדדית ומחויבות משותפת לאיכות - היא מקור השראה. אני גם יודע שהיא תמשיך להניע אותנו קדימה לחדשנות ולהצלחה.

אם יש משהו שלמדתי מהמהלך הזה, זה שביחד הכל אפשרי. אני גם יודע שזה לא הסוף ומחכים לנו עוד אתגרים מעניינים וגם הצלחות רבות!

מסעותיי עם הדרישות

"אי כתיבת מסמך דרישות הוא *הסיכון המיותר והגדול ביותר* שלוקחים בפרויקט תוכנה. זה מטומטם כמו לצאת למסע חציית הסהרה מצוידים רק בבגדים שלגופכם, בתקווה שתסתדרו איכשהו".

Joel Spolsky, Painless Functional Specifications

מי צריך בדיקות תוכנה?

את תחום דרישות התוכנה הכרתי כבר בפרויקט הראשון אליו הצטרפתי בתור בודק תוכנה. זה היה לפני כ-23 שנה, ואנחנו פיתחנו מודם לכבלים. חברות הכבלים בארצות הברית, מתוך רצון להבטיח שמודמים שיחוברו לרשתות שלהם אכן יעבדו כמו שצריך, הקימו תאגיד שהגדיר תקן מפורט עבורם (תקן DOCSIS). התקן הכיל כ-3500 דרישות מפורטות, והתאגיד סיפק טבלה מסודרת עם כל הדרישות. פעולה זו חסכה מאמץ מכל מי שעסק בפיתוח מודמים: במקום שכל ארגוני הפיתוח היו צריכים לזקק את הדרישות מהתקן בעצמם - כולל הטעויות ואי ההבנות - הם קיבלו רשימה מוכנה ורשמית. דרישות אלה הגדירו למפתחים במדויק את הפונקציונליות הנדרשת מהקוד, ועבורנו, הבודקים, מה בדיוק צריך לבדוק. מצד אחד, זה היה קאדר. לכל דרישה היה צריך לכתוב לפחות תסריט בדיקה אחד, ולעיתים רבות – יותר מאחד. מצד שני, זה היה נהדר. על מנת לעבור את תהליך ההסמכה (certification) המחמיר של תאגיד חברות הכבלים, היינו חייבים לבדוק כל פרט במוצר. העובדה שכל הפרטים הוגדרו היטב בדרישות עזרה מאוד למפתחים לא לשכוח משהו, ולנו, הבודקים, להיות בטוחים שבדקנו הכל.

> **"ההחלטה לנהל דרישות באופן מלא מחייב "leap of faith"**

בפרויקטים הבאים, בתחום אחר, המצב כבר היה שונה. אמנם היה מסמך דרישות שהגדיר דרישות כלליות, אבל השטן, כידוע, הוא בפרטים, ובהעדר פרטים התרבו הבאגים והוויכוחים אם משהו הוא בכלל באג. (אולי זה חוק קוסמי? "הסכום הכולל של מספר הדרישות המפורטות ומספר הבאגים בפרויקט הוא ערך קבוע"...).

אחרי ששחררנו פרויקט עם 12 גרסאות סופיות (כל פעם מצאנו עוד באג קריטי ברגע האחרון), ההנהלה החליטה שצריך להבין מה לשפר, ועלי הוטל להוביל את החקירה. המסקנה העיקרית הייתה שאנחנו צריכים לנהל דרישות בצורה יותר מקצועית. אחת מהאנליזות המשכנעות ביותר הייתה התפלגות בעיות השורש שגרמו לבאגים בחומרה גבוהה או קריטית. הנתונים הראו שכ-20% מהבאגים החמורים וכ- 36% מהבאגים שנסגרו כטעות ("not a bug") נבעו מבעיה בהבנת הדרישות, אי ידיעת הדרישות או דרישות מעורפלות. כשהסתכלנו על מספר כל הבאגים בפרויקט, היה מדובר על מאות באגים שאפשר היה לשייך אותם לבעיה בדרישות.

משם קצרה הדרך לפרויקט גדול של הכנסת כלי לניהול דרישות, הגדרת תהליכי עבודה סביב הדרישות, ובכלל השקעה בתחום. בקבוצות הבאות בהן עבדתי היו רמות שונות של ניהול דרישות, אבל אף קבוצה לא הגיעה לרמה של פרויקט מודם הכבלים – עד הפרויקט הנוכחי, ועל כך בהמשך.

למה להשקיע בניהול דרישות?

בתור בודק, התשובה פשוטה ומיידית. אם יש דרישות מפורטות, יש בסיס מוצק לניהול הבדיקות. "בסיס הבדיקות" מוגדר, והחיים טובים. זה אמנם לא פותר את הצורך להבין מה הסיכונים במוצר, במה להשקיע יותר ואיזה חלק במוצר יכול להסתפק במעט בדיקות, את מה לאטמט, מה שכחו בדרישות, וכו'. קצת אתגר בחיים בכל זאת צריך שיהיה. אבל לפחות יש מושג ברור מה המוצר אמור לעשות, ברמה שאפשר לבנות עליה.

בתור מפתח... הבעיה סבוכה יותר. תיאורטית, כולם מסכימים שיהיה נחמד אם רשימת הדרישות הסופיות למוצר תופיע לפתע משמים, ואם לא קשה לכם, כבר מוטענת לכלי ניהול שאין בו אף מגבלה. אבל לעשות את זה בעצמנו?

העניין הוא ככה: לניהול דרישות יש תקורה (overhead) לא קטנה. זה מתחיל בצורך להחליט באיזה כלי להשתמש; רק תהליך בחירת הכלי יכול לקחת חודשים. צריך לפרק כל דרישה של הלקוח או של מרקטינג להמון תתי פרטים ולכתוב אותם בצורה שאי אפשר יהיה שלא להבין. אחר כך להעלות אותם לכלי שבחרתם, תוך התקוטטות עם המגבלות שלו (תמיד משהו בכלי יהיה מעצבן; זה עוד אקסיומה קוסמית). צריך להשקיע זמן בסקירה אמיתית של הדרישות וכשכבר הכל כתוב וסרוק, ונראה לכם שזהו, מעכשיו זה רק שעה שעתיים בשבוע, מסתבר שלא. חתמתם קבע, ומידי יום יש שינויים שדורשים הוספה, עדכון או מחיקה של דרישות.

וכל זה קורה לפני שארגון הפיתוח יכול לראות איזושהי תועלת בכל ההשקעה. אמנם אנשים כמוני מבטיחים להם שבסופו של דבר המאמץ הזה יתבטא בפחות באגים, פחות אי הבנות ופחות צורך לקודד מחדש חלקים בתוכנה. אבל אם הם לא השתתפו אף פעם בפרויקט שניהל דרישות כמו שצריך, "לפי הספר", ההחלטה לנהל דרישות באופן מלא מחייבת את מנהלי הפיתוח לעשות "קפיצת אמונה" (leap of faith): לקיחת הסיכון של הקצבת זמן יקר לצורך ניהול הדרישות, ועמידה בפני הטענות של הצוות שכל הפעילות היא בזבוז משווע של משאבים. ולמה לקחת את הסיכון? הרי הם הצליחו בעבר לשחרר תוכנה גם בלי זה!

עדויות מהשטח

בשנה האחרונה, עיקר העבודה שלי מתמקדת בשני תחומים: הגדרת תהליכים וכתיבת הניירת הנדרשת לאישור של מוצר רפואי בארצות הברית, וניהול דרישות עבור המוצר. שני תפקידים שממלאים לי פחות או יותר את היום, כשניהול הדרישות לוקח בערך 50% מהזמן שלי. וכל זה על מוצר שהוא בעצם די קטן (אנחנו עומדים עכשיו על כ-600 דרישות; נגיע כנראה לאזור ה-800. תשוו את זה ל-3500 של מערכת מודם כבלים).

> **"יש לבודקים יכולת להשפיע על החלטה לנהל דרישות"**

האם זה שווה את זה? אני חושב שכן, אבל לא זו הסיבה האמיתית שכל הארגון מתיישר ומנהל דרישות. פשוט: על מנת לקבל אישור של ה-FDA לשווק מוצר רפואי בארה"ב צריך להדגים ניהול דרישות וניהול טבלאות מעקב בין הדרישות, הקוד, הסיכונים במוצר, הבדיקות ותוצאותיהן.

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

ובכל זאת, עניין אותי מה חברי לצוות חושבים על ניהול דרישות והאם הם מזהים תועלת בזה. ערכתי סדרה של פגישות קצרות עם מנהלי קבוצות הפיתוח והבדיקות בפרויקט, ושאלתי אותם ארבע שאלות:

- האם כבר עבדתם בעבר בפרויקט שבו הדרישות נוהלו כראוי? (השאלה נועדה להבין אם הם חוו את שתי האפשרויות ולכן יכולים להעיד מה עדיף)
- מה היתרונות בניהול דרישות?
- מה החסרונות?
- אם הייתם מנהלי הנדסה של פרויקט חדש, וההחלטה אם לנהל דרישות בצורה מלאה או לא הייתה בידכם, מה הייתם בוחרים?

בגדול אפשר לסכם את התשובות כך: כולם משוכנעים שיש ערך לניהול מפורט של דרישות. "זה מגדיר מה צריך לעשות"; "מקל על תהליך תכנון הספרינטים"; "משפר את התקשורת בין קבוצות הפיתוח ומזהה תלויות"; "עוזר להבין את הרציונל שמאחורי הדרישות"; "מקל על הגדרת מינימום הבדיקות שנדרש על מנת להגיע לכיסוי של התכונות של המוצר". אהבתי במיוחד את מה שאמר מנהל הבדיקות על פרויקט קודם שבו לא נוהלו דרישות: "זה היה תוהו ובוהו בסגנון חופשי".

המרואיינים הצביעו גם על חסרונות: יש תקורה רצינית. חייבים שיהיה מישהו שתפקידו הרשמי הוא לנסח דרישות, להוביל פגישות סקירה ולדאוג לרישום ועדכון הדרישות. בלי זה אין סיכוי שזה יעבוד טוב (נחמד לדעת שהם חושבים שאני לא מבזבז את הזמן). כמו כן, בגלל שכל הדרישות רשומות, הן חייבות להיות מעודכנות בזמן, כי כולם סומכים על זה. למשל: אם משנים משהו ולא מודיעים על זה לאנשי הבדיקות, הם ימשיכו לבדוק את הדברים הלא נכונים (המממ... לקחתי לתשומת ליבי לא לצבור backlog ארוך מידי).

עוד הסכמה הייתה שאם הם היו הקובעים בפרויקט, הם היו בוחרים להשקיע את המאמץ הנדרש ולנהל דרישות. חלק ציין שהיו בוחרים לעשות זאת אחרת מבחינת הכלי ו\או התהליך.

מעבר למשוב מהחברים, גם לי יש כמה הארות בעניין, אחרי עבודה של כשנה עם דרישות:

- צריך להחליט לאיזו רמת פירוט יורדים בכל תכונה של המוצר ולשמור על אותה רמה פחות או יותר בכל הדרישות. למשל: אנחנו שומרים קבצי וידיאו, ומידע על הקבצים עצמם (מטה-דאטא). האם לרשום דרישה אחת ("המערכת תשמור מטה-דאטא עבור קבצי הוידאו" ולפרט את רשימת הפרטים - ויש 20 כאלה בתיאור הדרישה) או לרשום 20 דרישות נפרדות ("המערכת תשמור את הרזולוציה של הקובץ"; "המערכת תשמור את אורך הקובץ בשניות" וכו' וכו').
- חשוב להחליט על סגנון השפה בה משתמשים לכתיבת הדרישות, וגם בזה לשמור על אחידות. זה מגביר את הקריאתיות של הדרישות ועוזר בניסוח שלהם. ספציפית, אני כותב את הדרישות על פי שיטה שנקראת EARS (מראה מקום – בסוף הטור).

"אם יש דרישות מפורטות, יש בסיס מוצק לניהול הבדיקות"

- צריך להחליט איך מנהלים דרישות של ממשק גרפי. אנחנו כותבים דרישות שמתארות את שדות הקלט וגם מצרפים את הגרפיקה (wireframe). יתכן שאפשר להסתפק רק בגרפיקה ולהניח שהמפתחים מסיקים מזה את השדות שיש בקלט. אבל זה משאיר שאלה פתוחה לגבי ערכים מותרים, אורך הקלט החוקי, מה התגובה לערכים לא חוקיים וכו' – שצריך לפרט איפשהו.
- קשה לבצע סקירה על דרישות שימומשו רק בעוד כמה חודשים. הסקירה יעילה הרבה יותר אם מתרכזים בדרישות שרלוונטיות לספרינטים הקרובים. בשלב זה יש למפתחים הרבה יותר מוטיבציה לעשות סקירה איכותית – הרי הם צריכים לדעת מה בדיוק רוצים שיקודדו. זה גם מתאים יפה לעקרונות של פיתוח אג'ילי.
- יעילות כלי ניהול הדרישות היא חשובה. כלי איטי מבזבז זמן, מוציא את כולם מהכלים, גורם להם לשנוא את התהליך ולנסות לעגל פינות. תנו עדיפות לכלי עם התממשקות יעילה לאקסל. אנחנו משתמשים בג'ירה, ולדעתי זה לא רע. לא כולם מסכימים איתי...

מה הבודקים יכולים לעשות?

ההחלטה אם לנהל דרישות אינה בידיים שלנו (הבודקים), אבל אנחנו כן יכולים להשפיע על זה. למשל: אפשר לחזור על האנליזה שעשיתי ולראות אם גם אצלכם הרבה מהבאגים נבעו מחוסר ניהול מדוקדק של דרישות. אפשר לצטט את העדויות של חברי לפרויקט, שהם אמנם אנקדוטליות אבל אני מקווה שהם בכל זאת שוות משהו. אפשר להסתייע במאמרים שבסוף הטור ואפשר גם (לא רעיון שלי, אבל אני לא זוכר איפה קראתי את זה) להתחיל לרשום לעצמכם באופן פיראטי את הדרישות – עד כמה שאתם מבינים אותן – לקומפוננטה או התכונה שאתם אחראים לבדוק. אחרי כמה זמן אנשים יתחילו לשים לב שאצלכם העניינים מתנהלים טוב יותר, ואולי זה ישכנע אותם לעבור לניהול דרישות מסודר.

תרגול נוסף בנושא:

1. EARS: Easy Approach to Requirements Syntax
https://www.slideshare.net/TechWellPresentations/bw6-terzakis
2. אם מחקרים עושים רושם על ההנהלה שלכם, אפשר לצטט מחקר של ג'יימס מרטין. המקור כנראה בספר שהוא כתב (אין לי אותו), אבל יש ציטוט במאמר של רוברט בנדר:

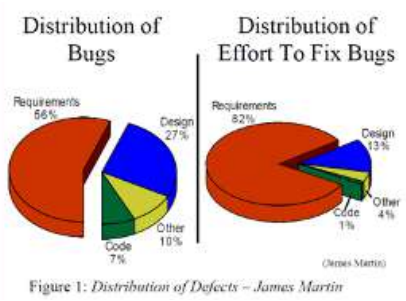
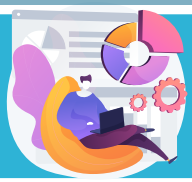


Figure 1: *Distribution of Defects – James Martin*

http://benderrbt.com/Bender-Requirements%20Based%20Testing%20Process%20Overview.pdf
Source (most likely): An Information System Manifesto, James Martin

3. קארל וויגרס (Karl Wiegers) כתב כמה ספרים קריאים בתחום.
4. ספר שממש מהנה לקרוא:
Exploring Requirements: Quality before Design
By: Weinberg, Gerald M., Gause, Donald C.
5. סדרת המאמרים המשעשעת (ובעלת הערך) של ג'ואל ספולסקי על דרישות:
https://www.joelonsoftware.com/2000/10/02/painless-functional-specifications-part-1-why-bother

משולחנו של שביט - הניתוח הצליח החולה מת



בכתבה זו אני ישר אגש לסיפור ואשתף אתכם במקרים אמנם נדירים אבל כאלו שעלולים לקרות וגם קורים מידי פעם.

בד"כ כשעובדים בסקוואדים - כל צוות מכיל מספר בודקים וכחלק מהעבודה, יתכנו שבועות שבהם צוות כזה ישחרר משהו כמו..5-6 גרסאות ולפעמים 3-4 גרסאות ישוחררו ביום אחד.

אבל מה קורה במצב שבו יש גרסה גדולה עם רמת סיכון גבוהה יותר?

ולמרות שזה עובר QA אנחנו יודעים שקשה מאוד תמיד לעלות על כל דבר קטן. במיוחד שעל גרסה חדשה טרם נכתבה אוטומציה מלאה. במקרה הממש טוב – עובדים על זה אבל גם זה רחוק מלהיות יציב ושלם.

ואשאל אתכם יותר מזה...מה קורה כשגרסה גדולה עולה לאחר שעברה אותנו, עברה QA ובצוות אחר גם הגרסה עברה את ה-QA עם כל הבדיקות הנדרשות.. אוטומציה סרקה בכל גרסה מה שצריך כדי לוודא שהכל תקין.

אבל ...אז קורה שבאותה נקודת זמן הועלו 2 גרסאות שונות – איך קרה ששתי הגרסאות הועלו באותו זמן? – שתיהן נבדקו על סביבות QA שונות, שתיהן הגיעו משתי צוותים שונים.

התברר שבשתיהן היו קונפליקטים שהתגלו רק לאחר העלייה לאוויר וגרמו ל-3 באגים חמורים ביותר במערכת.

מה עושים עם תפוח האדמה הלוהט הזה שהגיע אלינו?..

טוב, אז כמובן שבגרסאות גדולות כאלו לא לוקחים סיכונים מיותרים ומבצעים ישר Rollback שזה חזרה לאחור לגרסאות קודמות. אממממה.. במקרה הזה, היה צורך לבצע double rollback כי לא הצליחו להבין מה ה – rout cause של אותן בעיות באותו זמן ומאיזו גרסה הגיעו הבעיות האלו.

לאחר חקירה הבנו ששילוב שהיה די מסוכן בין 2 הגרסאות האלו יצר קונפליקט שבתצורה הקיימת לא היה ניתן לעלות על אותן בעיות מבלי ליצור תהליך מסודר יותר. כלומר, לוא היינו עובדים בתהליך מסודר יותר, היינו מגלים את הבעיות הללו בשלב מוקדם יותר.

לדוגמא,– להעלות גרסה one by one, לבצע rebase למאסטר בגרסה שאנחנו בודקים. כלומר, ליישר את סביבת הבדיקות לפרודקשן ואז להמשיך לבדיקות.

וזאת למרות שמדובר בצוותים שונים שמעלים גרסאות כל הזמן כי הצוותים עלולים לגעת בחלקים שונים במוצר שיהיו קשורים גם ל"שטחים" של צוותים אחרים שעובדים על אותו מוצר.

בסוף, הבעיות טופלו. הגרסאות נבדקו כמו שצריך והכל עלה ללא בעיה נוספת.
אבל מה שכן עלה...עלה לחברה ב-לא מעט כסף למרות שה-Rollback נעשה די בזמן ולא היה חי יותר משעתיים.

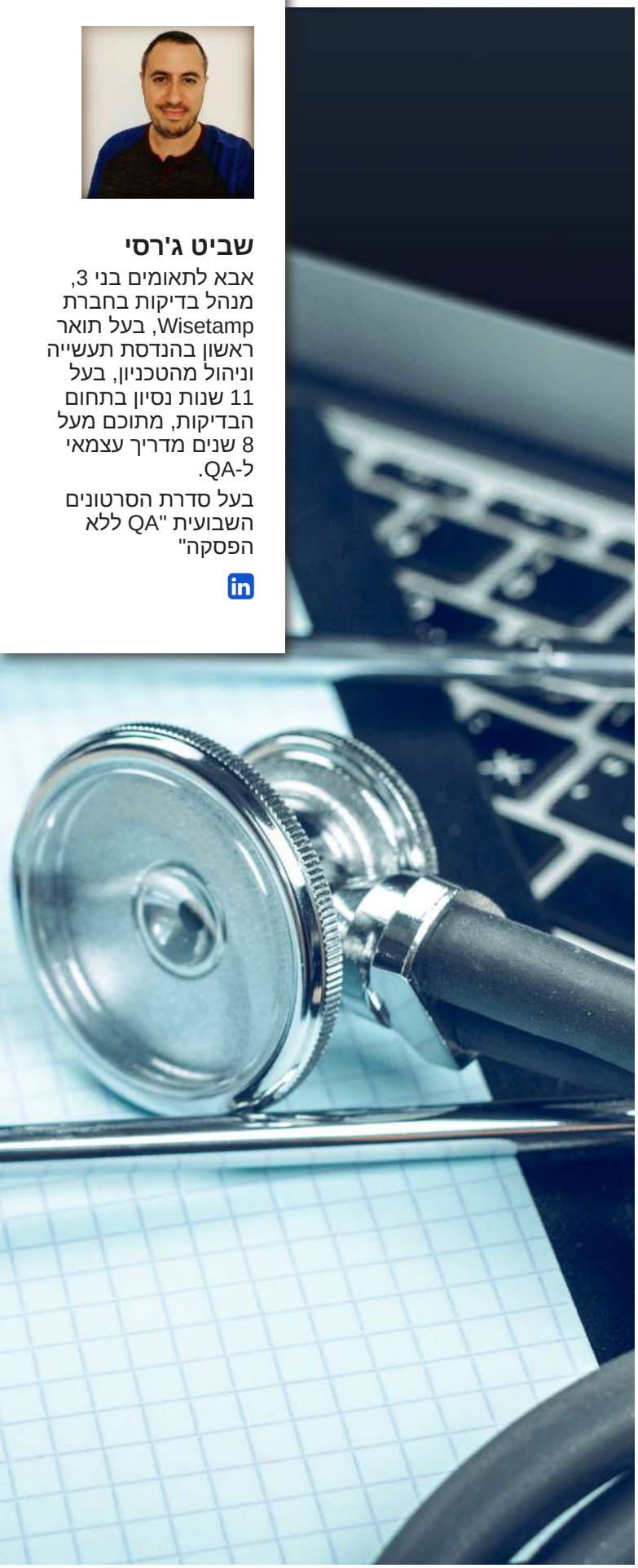
והתובנה מכל הסיפור הזה... - שימו לב!!!
היו ערניים לגרסאות שהן חריגות בגודלן ורמת הסיכון שלהן גבוהה יותר. זה חשוב ביותר. יותר מודעות בקרב המפתחים וגם בקרב הבודקים.

מה יוצר את המודעות לזה? -
תקשורת! חברים...ת-ק-ש-ו-ר-ת.

צריך גם לשקול code freezes , (בכדי) שכאשר יתגלו בעיות יהיה ניתן להגיע לשורש הבעיה בצורה קלה יותר.
צריך לחשוב על תהליך עבור שחרור גרסאות חריגות בגדלן.
אולי אפילו לנהל את הסיכונים כתהליך מתודי.

העלאת הערנות והמודעות במהלך רוטינות של הפיתוח ושל הבודקים.
גם תקשורת ותיאום והכנה שיכול לקרות מצב שנגיע ל – rollback.
הניתוח הצליח, אבל החולה מת. ועם זאת.. זכרו – "ספינה הכי בטוחה כשהיא עוגנת בנמל, אך לא לשם כך נועדו ספינות...."

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



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"

מהפרונט לבק של עולם הבנקאות- אילנה רז

שמי אילנה רז נשואה עם שלושה ילדים, גרה במודיעין, עובדת כחמש שנים כבודקת תוכנה, לפני כן עבדתי כ-17 שנה כבנקאית במגוון תפקידים בבנק יהב. בשבתות אני אוהבת לרוץ/ללכת (למרות בשנתיים האחרונות אני בעיקר הולכת בגלל פציעה), לעשות טיול ארוך עם הכלב (אם זה מתאפשר אז גם באמצע השבוע). אוהבת לעשות יוגה ופילאטיס, לטוס לחו"ל, לשמוע מוזיקה, לראות הופעות, להתפנק בבית עם פופקורן וסרט טוב עם כל המשפחה, לשמוע פודקאסטים.

תחילת דרכי בבנקאות

איך הגעתי להיות בנקאית, בשנת 2001 סיימתי תואר ראשון במדעי החיים. בתחילת התואר רציתי לעסוק במדע ובמחקר, אבל בסיום התואר עמדו בפניי שתי אפשרויות: להמשיך לתואר השני או ללכת במסלול שחברותיי לתואר המשיכו - להיות תעמולנית רפואית בחברת תרופות. להמשיך לתואר שני לא רציתי אז הלכתי לראיונות למשרות בתעמולנות רפואית ומהר מאוד הבנתי ששיווק תרופות בבתי חולים לא ממש מושך אותי.

הבנתי שאני צריכה להחליט מה אני רוצה לעשות. בזמן שאני חושבת על העתיד שלי התחלתי לחפש עבודה ומצאתי את עצמי עובדת בבנק יהב בירושלים. לאחר 6 שנים של עבודה בבנק במחלקת קופות-הגמל וקרנות ההשתלמות החלטתי לעשות שינוי ועברתי לעבוד בסניף חדש בעיר שבה אני גרה. ללא כל ניסיון בעבודה סניפית בקבלת קהל, התחלתי בדרך חדשה בפתיחתו של סניף חדש במודיעין. די מהר הבנתי שאני עושה את מה שלא רציתי לעשות לפני 6 שנים, כשהגעתי לבנק וזה היה שיווק. מהות העבודה בעיקרה הייתה להביא לקוחות לפתיחת חשבון, להביא לקוחות לקבל הלוואה, לפתוח תוכניות חיסכון, פיקדונות, להוציא עוד כרטיסי אשראי.... ואז למעשה הבנתי שאני עוסקת בשיווק וזה לא מה שרציתי לעשות, לא אחרי התואר ולא בעבודתי בבנק.

הזדמנות ראשונה להיות בודקת תוכנה בבנק

איכשהו הנוחות של הבנק לא ממש מצליחה לכבות את הרצון העז שלי לעשות משהו משמעותי יותר עבורי. לכן ניסיתי לשנות כיוון, למדתי הוראה, מלא משמעות, לא? אבל אחרי כמעט שנה של לימודי הוראה, הבנתי שזה לא מתאים לי.

במקביל, קרה משהו בבנק, הוחלט בהוראת בנק ישראל להחליף את מערכת הליבה של הבנק, שהייתה תלויה בבנק פועלים, למערכת חדשה לגמרי, ובעקבות השינוי במערכת הליבה של הבנק, נפתחו משרות של בודקי תוכנה בתוך הבנק, תפקידם לבדוק את מערכת הליבה החדשה שפותחה ע"י חברת TCS.

הגשתי מייד קורות חיים ולא התקבלתי, אחרי כמה חודשים נפתחו עוד משרות לבודקי תוכנה ושוב הגשתי מועמדות, שוב לא התקבלתי, בדיעבד הבנתי שמנהל הסניף בו עבדתי לא רצה להמליץ, כי רצה שאשאר בסניף, אבל בסוף, הכל הסתדר לטובה. בסופו של דבר הגעתי לעבוד בפרויקט של הסבת המערכת הסניפית מהדלת האחורית, אבל לא למשרה שרציתי.

אחרי יותר משנה בעבודה על הפרויקט, שוב החזירו אותי לעבוד בסניף (תיגבור הסניף לאחר ששתי עובדות יצאו לחופשת לידה)

ואז החלטתי שאני לא ממשיכה יותר בבנק, שכחתי את הרצון שלי להיות בודקת תוכנה, חברה המליצה לי לעשות קורס דאטה אנליסט. החלטתי שאני עושה את הקורס ובמקביל לקורס הגשתי מכתב פיטורים לבנק.

הזדמנות נקרית בדרכי ואני זורמת איתה – הפעם הראשונה

החלטתי לעבוד בבנק עד סיום הקורס ולאחר מכן לעזוב את הבנק ולחפש משרת ג'וניור כדאטה אנליסט. חודש לפני סיום הקורס, חברה שעבדה איתי בסניף, התקשרה, ואמרה לי שהבנק פרסם משרה של בודקת תוכנה (אחת הבודקות קודמה לתפקיד ניהולי אחר בבנק).

החברה אמרה לי שאני חייבת להגיש מועמדות. להבדיל ממני, היא זכרה את הימים בהם ממש רציתי להיות בודקת תוכנה בבנק, היא שכנעה אותי להגיש מועמדות למשרה. הגשתי מועמדות בלי רצון של ממש, הרי כבר ידעתי שעוד רגע אני עוזבת את הבנק לטובת מקום חדש בתפקיד חדש של דאטה אנליסטית, כמובן באופן בלתי צפוי בעיני, זימנו אותי לראיון אצל מנהל המחלקה והתקבלתי. ככה התחלתי לעבוד כבודקת תוכנה, ללא הכשרה או קורס, בלי ידע של ממש מה אני עושה, רק עם חפיפה קצרה על Bugzilla, ו-QC.

הייתי צריכה ללמוד לבד, על מסמכי אפיון, באגים, בדיקות, כתיבת באגים והרצה של טסטים.

במשך 3 שנים של הרצת תסריטים, פתיחת באגים על המערכת של האתר, האפליקציה הבנקאית, בדיקות נגישות, בדיקות אימות לבאגים שתוקנו. הרגשתי שמיציתי, לא את מה שאני עושה, אלא את המקום, הרגשתי שאני ובנק יהב צריכים להיפרד...

"זימנו אותי לראיון אצל מנהל המחלקה והתקבלתי. ככה התחלתי לעבוד כבודקת תוכנה, ללא הכשרה או קורס, בלי ידע של ממש מה אני עושה, רק עם חפיפה קצרה על Bugzilla, ו-QC"

הזדמנות נוספת בדרך שאני לוקחת – הפעם השנייה

הרגשתי שמיציתי את העבודה בבנק, אבל עדיין ההרגשה הזאת לא ממש התממשה לה, כלומר הרגשתי את המיצוי, אבל לא עשיתי עם זה כלום. הימים הם שלהי הקורונה סוף שנת 2021, אני בדיוק חוגגת 20 שנות עבודה בבנק יהב, מטורף!!!

ושוב חברה שעובדת איתי במחלקה, ראתה מודעה בלינקדאין, על בנק דיגיטלי חדש שמחפש בודקת ידנית עם ניסיון בבדיקות מובייל, וכמובן שכנעה אותי להגיש מועמדות. אחרי יום של שהייה, רעננתי את קורות החיים שלי, הסתכלתי על המשרה בלינקדאין וראיתי שעובד שם מישהו שעבד איתי פעם בסניף של בנק יהב במודיעין, מייד פניתי אליו, אחרי שהוא המליץ לי על הבנק, הבנתי שאני רוצה ויכולה להתקבל, שלחתי קורות חיים.

אחרי ראיון טלפוני קצרצר, ראיון פנים מול פנים עם מנהל הבדיקות, והמנהלת שמעליו, ראיון עם HR התקבלתי.

התקבלתי בזכות הניסיון שלי בבנק יהב במערכת הליבה הבנקאית, שהייתה זהה למערכת הליבה בבנק הדיגיטלי, כמובן גם בזכות

מהפרונט לבק של עולם הבנקאות- אילנה רז

הניסיון שלי כבודקת, אבל הניסיון שלי, כבודקת תוכנה התאים יותר לעבודה בבנק יהב, ולא בבנק One Zero, כי העבודה בבנק וואן זירו כבודקת תוכנה, הייתה שונה לגמרי. פתאום הרגשתי שאני נמצאת במגרש של הגדולים, בהיי-טק. למרות ש-One Zero הוא בנק אבל העבודה בו דומה יותר לעבודה בחברת היי- טק.

ממש התלהבתי מהמקום החדש שהתחלתי לעבוד בו, הרגשתי סוף כל סוף שאני במקום שאני יכולה להאמין בו, הקונספט שלו היה שונה מכל בנק אחר, זה לא עוד בנק שרוצה למכור ללקוחות הלוואות, פיקדונות כרטיסי אשראי ועוד אשראי, אלא באמת לנסות להיות בנק תומך ולומד את הלקוח ואת החשבון של הלקוח ולתת לו הצעות ערך שהם מתאימות לו לטובתו. ההתלהבות שלי כנראה הייתה ניכרת עוד בראיון עם ה-HR כי סיפרתי לה כמה אני אוהבת דברים טכנולוגים חדשים, לא מפחדת מקדמה, ושבאתי מבית שתמיד התקדם עם כל החידושים, אפילו אבא שלי, למרות גילו, תמיד רצה ללמוד דברים טכנולוגים חדשים, ולא פחד מקדמה.

"התקבלתי בזכות הניסיון שלי בבנק יהב במערכת הליבה הבנקאית, שהייתה זהה למערכת הליבה בבנק הדיגיטלי, כמובן גם בזכות הניסיון שלי כבודקת"

אבל עם כל ההתרגשות וההתלהבות, עמדתי מול אתגר גדול, איך אני מצליחה להשלים את כל הפערים שיש לי כבודקת, ללא הכשרה פורמלית, ללא ניסיון רב, בכתיבת תסריטים, בהכרות עם כל סוגי הבדיקות, הייתי צריכה ללמוד מושגים טכנולוגיים ומושגים בסיסיים בהייטק, להכיר תוכנות חדשות הרגלי עבודה חדשים המקובלים בהיי-טק. אג'ייל, סלאק, ג'ירה וכו. בקיצור המון דברים שצריך ללמוד. הצלחתי להתגבר על הכל. איך זה קרה? פשוט, ישבתי ולמדתי, ולמדתי הרבה, שאלתי המון שאלות את חבריי לצוות, ואנשים מצוותים אחרים, שאלתי הרבה, הסתכלתי על דוגמאות, דפי קונפלואנס, סרטונים ביוטיוב, ועוד ...

רק אחרי ניסיון של כשנה כבר הייתי עם מספיק ביטחון עם כל מה שעשיתי. שנה וחודש אחרי שהתחלתי את העבודה, המנהל שקיבל אותי פוטר. פתאום הבנתי שככה זה בהיי-טק, שזה לא בנק, אין קביעות, אין ועד עובדים, יש הנהלה שהחליטה על צמצום, וכמה ימים אחרי, אנשים מפוטרים.

"מכל ראיון שעברתי למדתי, למדתי מהן החוזקות שלי ומהן החולשות שלי"

וככה בדיוק זה קרה גם לי, לאחר שנה וארבע חודשים היו צמצומים והתפקיד שלי בוטל.

הייתי ממש בשוק, לא עיכלתי, התכחשתי, הייתי ממש אובדת עצות... מה אני עושה?

דיי מהר עדכנתי את קורות החיים שלי ועדכנתי סטטוס בפרופיל הלינקדאין והתחלתי במסע חדש של חיפוש עבודה, אבל לא כי מיציתי, אלא כי לא הייתה לי ברירה.

בינתיים בוואן זירו האריכו לי את התאריך שבו אני מסיימת את עבודתי, בחודש, ואז שוב האריכו לי ושוב האריכו לי... בסה"כ המשכתי לעבוד עוד 8 חודשים אחרי התאריך המקורי.

קל זה לא היה, לעבוד ולנסות לתת 100% לטובת מקום שצמצם אותי, לעבוד כשבעצם אני יודעת שאני לא נשארת, שאני יודעת שאני מפוטרת. לא קל, במקביל לחפש עבודה, ראיונות, מבחני בית, עוד ראיונות, והרבה הרבה לא, לא תודה, אנחנו ממשיכים עם מועמדים מתאימים יותר. מכל ראיון שעברתי למדתי, למדתי מהן החוזקות שלי ומהן החולשות שלי. גיליתי למרות שעבדתי כמעט שנתיים בהיי-טק שחסר לי עוד הרבה ידע, ידע טכני, ידע בבדיקות ובמושגים. אבל לא ויתרתי, המשכתי ללמוד, להגיש עוד מועמדויות, והמצב בארץ לא קל, הייתה המהפכה המשפטית, ההפגנות, ומלחמה שטרפה את כל הקלפים.

אז עכשיו זהו המצב, אני מסיימת את עבודתי בוואן זירו, אני עדיין מחפשת עבודה, עדיין לומדת, אני עדיין משפרת קורות חיים, עדיין משפרת את הידע שלי ובעיקר מנסה להיות חיובית ואופטימית.

כן אין ברירה, חייבת להיות באנרגיות חיוביות, למרות שעזבתי מקום בטוח, והגעתי למקום חדש ולא ידוע. למרות שהתגברתי על כל קושי שהיה לי במקום החדש, למרות שפוטרתי, ושלא התקבלתי עדיין לשום משרה, למרות שאני עדיין צריכה ללמוד הרבה, למרות שאני עומדת מול עתיד לא ידוע, מבחינה מקצועית, כלכלית, ביטחונית, הראש תמיד מורם...

עושים QA לקריירה
מה כדאי (ומה לא כדאי), בכניסה לתפקיד חדש

התחלת תפקיד חדש? התקדמת בארגון הקיים או שהתחלת תפקיד ניהולי - זה בשבילך.

תפקיד חדש ובוודאי שבארגון חדש הוא אקט של הסתגלות קיצונית. גם לאלו מבניכם שרגילים לשינויים, או שכבר עברו את זה כמה פעמים, יש אתגר משמעותי בלהתמקם נכון מול ההנהלה והקולגות, לייצר הצלחות ולבוא לידי ביטוי. זה שצריך ללמוד את המוצר, הארכיטקטורה והטכנולוגיה זה ברור. כאן תקראו על השאר.

בכל תפקיד חדש, מתקיים מפגש משולש בין הציפיות שלנו מעצמינו, הצפיות מהסביבה, וציפיות הסביבה מאיתנו. ההתנהגות והחוויות שלנו בחודשים הראשונים יאפיינו את האיזון בין שלושת אלו ויעמדו בבסיס כל מערכות היחסים שנבנה בארגון.

זה לא משנה אם אתם QA, מפתחי תשתיות או מנהלי קבוצות, זה לא משנה אם זה התפקיד הראשון או החמישי שלכם. עדיין תחוו את הדינאמיקה באותו משולש ציפיות וההתנהלות שלכם בחודשים הראשונים תעצב את איך שאתם תתפסו את עצמכם ואיך שאחרים יתפסו אתכם, חודשים רבים קדימה.

מה כדאי לעשות בשלושת החודשים הראשונים בתפקיד?

בכל תפקיד מתחילים מהמאקרו אל המיקרו ומבפנים החוצה.

מאקרו – אתם בארגון:

תלמדו את התחום בו הארגון פועל ותבינו מהם מדדי האיכות והיעדים של המנהל/ת שלכם.

אם המוצרים הם B2B, תשאלו מי הלקוחות האסטרטגיים או הגדולים ביותר, איך איכות נמדדת בעיניהם ולמה הם מצפים? כשתתקלו בבאג, או באירוע פרודקשן, תוכלו להבין את העדיפויות וכמה ללחוץ לפתרון. אם המוצר הוא B2C תשאלו - מה הכי ישפיע על תפיסת הלקוח את איכות המוצר?

כשתדעו את אלו, תקבלו כלים להשפיע בארגון.

מיקרו - אתם בתפקיד:

תוודא/י שיש לכם Buddy מהצוות שאתם חלק ממנו, גם אם אתם בתפקיד ניהולי. אם לא הוקצה אחד כזה, תבקשו. Buddy הוא מישהו שהוגדר לו שחלק מתפקידו, הוא לעזור לכם, להיות זמין בשביל שתוכלו להתאקלם ביעילות ולענות על השאלות שקשורות לאיך דברים בפועל מתנהלים ...

לאורך 20 שנות הקריירה שלי, עברתי בין תפקידים ובין ארגונים עד שלפני כמה שנים הגעתי להיות ה-Head of Quality בארגון גדול שמאוד התלהבתי ממוצריו ותחומי האחריות שהולכים להיות לי.

כמה ימים לאחר שהתחלתי את התפקיד, בישיבה עם כל הנהלת הארגון, תוך כדי שמנכ"ל החברה מציג אותי ואת תפקידי כסמנכ"לית, אמר ה-VP R&D מול כולם שהוא לא חושב שהתפקיד הזה נחוץ בכלל ושהוא כבר מינה מנהל איכות מטעמו. הסוף היה טוב, אך ההתחלה העבירה אותי שיעור חשוב.

אם אתם נכנסים לתפקיד ניהולי - תפעלו כדי לזהות מוקשים מראש. תוכלו לשאול את המנהלים שלכם - איך בעיניהם נמדדת הצלחה בתפקיד ומה המיקוד שצריך להיות לכם בשלושת החודשים הראשונים. תוכלו לשאול מה עלול להכשיל אתכם בתפקיד. (לעיתים יש היסטוריה שכדאי להכיר) מי ממשק העבודה הכי מאתגר ומאותגר שהם צופים שיהיה לכם ולמה הוא זקוק?

אם לא קיבלתם תשובות טובות, תשאלו את חברי הצוות שקיבלתם - הם ידעו את רוב התשובות.

> **"בכל תפקיד חדש, מתקיים מפגש משולש בין הציפיות שלנו מעצמינו, הצפיות מהסביבה, לציפיות הסביבה מאיתנו"**

בשבועות הראשונים בתפקיד, טבעי שתשימו לב לכל מה שלא עובד טוב. כשאנחנו מגיעים מבחוץ, קל יותר לשים לב לשגרה האוטומטית שאחרים עובדים בה. אתם תזהו תהליכים לא יעילים או מתודולוגית פיתוח בדיקות חסרה ואז יעלה בכם דחף לשנות את זה מיד או לכל הפחות להציף את זה למנהלים שלכם. אז חכו - יהיה נכון לעשות את זה לאחר שפגשתם את כל מי שצריך לפגוש וחוויתם איך ספרינט/תהליך שחרור גירסה קורה. בינתיים, תמפו לעצמכם את הפערים שאתם נתקלים בהם וכדי להבין מה הכי דחוף לשנות, תשאלו את עצמכם מה המחיר שהארגון משלם על חוסר זה? זמן, יעילות, בטחון נמוך באיכות הגירסה, שחיקה של חברי הצוות.

שינוי מתחיל ממוטיבציה. זה יעזור לכם לשכנע בפתרון שתציגו וזה יתרום למיצוב חיובי שלכם בפני ההנהלה. להגיע עם בעיות ופערים בלי הצעות לפתרון יעשה אגב, את ההפך.

אחרי שלמדתם את הארגון והשפעת האיכות במאקרו, ועברתם לזהות את מאפייני סביבת התפקיד שלכם במיקרו, תשתמשו בכלי הבא כדי לחדד את האינטואיציה שלכם.

> **"אם אתם נכנסים לתפקיד ניהולי - תפעלו כדי לזהות מוקשים מראש"**

מפנים החוצה:

בכל תפקיד קריטי להשתמש בכלי הניהולי הזה ולהכיר את עצמכם לעומק. להבין אילו חוזקות הולכות לעזור לכם בפרק הזה בקריירה, למה אתם זקוקים כדי להצליח בו ומה כבר למדתם על עצמכם בעבר וכדאי שתשימו לב אליו גם הפעם (גיליון 31 - רטרוספקטיבה להצלחות).

תענו על השאלה, למה (Why) בחרתם בתפקיד הזה? למשל - הוא מהווה התפתחות, כי כשלא טוב לי אני זז/ה וכו'.

מה חשוב לי שיבוא לידי ביטוי בתפקיד? למשל - ללמוד טכנולוגיה חדשה, יכולת השפעה, עצמאות בקבלת החלטות וכו'.

ואם זה תפקיד ניהולי אז מהם העקרונות המנחים אותי בבניית הצוות/ים שלי? למשל - ערבות הדדית, למידה מתמדת, שקיפות וכו'.

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
מה כדאי (ומה לא כדאי), בכניסה לתפקיד חדש

ולבסוף - מה אתם יודעים בוודאות על עצמיכם שעוזר לכם כשקשה? למשל - שיש לי למי לפנות, שיש לי תמיד אפשרות לבחור אחרת וכו'.

התשובות לאלו הן מעין עוגן פנימי, מקום לחזור אליו/ בכל פעם שיש פער גדול בין שני קודקודי משולש הציפיות - שלך מעצמך, שלך מאחרים ושל אחרים ממך. לענות על השאלות האלו בכניסה לתפקיד יאפשרו לכם ויסות פנימי תוך כדי שאתם מגבשים חוסן.

הגדרת החוסן הטובה ביותר בעיני היא זאת "לנהל את עצמי, את העולם הפנימי שלי כשהעולם בחוץ לא תואם את הציפיות שלי". כשאתם נכנסים לתפקיד חדש, אתם נכנסים לחדר כושר לחוסן. ככל שהתפקיד בכיר יותר, כך נדרש ניהול פנימי משוכלל יותר. לפעול מפנים החוצה, אומר להכיר את מה שחשוב לנו ועוזר לנו להצליח ולהגביר מודעות ובכך להתנהל מתוך בחירה.

ד"ר אייל דורון, חוקר ומפתח חשיבה יצירתית, אומר שכשאנו נכנסים לתפקיד חדש, לארגון חדש, אנחנו לא נכנסים אליו לבד. אנחנו מגיעים עם הקשרים שלנו, קולגות מקצועיים להתייעץ איתכם, מנטורים שהיו לנו בעבר, ובכלל - אנשים שיכולים לעזור לנו לצלוח את הדרך חכם. כדאי לדאוג שיהיו כאלו בסביבתכם, קולגות מקצועיים שאתם סומכים עליהם, כאלו שהיו בתפקיד דומה שיוכלו לשקף לכם את מה שאתם לא שמים לב אליו ויתמכו בכם בתהליך ההתבססות החשוב הזה בתפקיד ובארגון.

לכל אחד יש סיפור על החודשים הראשונים שלו בתפקיד, אלו יכולים להיזכר אצל כל אחד כחוויה חיובית או ככזאת שמלמדת שיעור חשוב. במצבים המצריכים הסתגלות קיצונית, כמו בכניסה לתפקיד חדש, קריטי לפעול מהמאקרו למיקרו ומבפנים החוצה. כשאתגרים גדולים יגיעו - אתם תדעו איך נכון לכם להתקדם ובעבור מה אתם עושים את זה באמת.

ITCB Israeli Testing Certification Board

ISTQB International Software Testing Qualifications Board

היו הראשונים להשתמש!!!

מילון המונחים הרשמי של ISTQB® בהשקה רשמית ומשותפת עם ITCB® לשפה העברית!!!

חיפוש מונחים, תרגום מונחים בין שפות שונות, שמירה לקובץ, הדפסת מונחים ועוד.

לפרטים נוספים למידע והסברים נוספים הכנסו לאתר ITCB בקישור

bit.ly/ISTQB_Glossary

אל תפחדו מקוד ספגטי בתוך פרויקט אוטומציה אלכס קומנוב

לפני כמה שנים התחלנו (בחברת SeatGeek) את פרויקט האוטומציה (Playwright + Typescript) הניסיוני שלנו. עם חלוף הזמן הוא הפך להיות הפרויקט הראשי שעובד עד היום עם אלפי טסטים מכל מיני סוגים וכולל הרצה על מגוון גדול של סביבות. הוא מתפרס על פני עשרות אלפי שורות קוד שהתחילו כרעיון ראשוני עם כמה שורות קוד בודדות וכמה טסטים פשוטים.

לאורך הזמן, ביצענו פעמיים עבודות תחזוקה מאוד רציניות בנוגע לפרויקט.

- פעם אחת על מנת לתקן המון טסטים נופלים.
- פעם שנייה בשביל להטמיע תשתית חדישה יותר.

אם להסתכל על כל הדרך שעבר פרויקט האוטומציה, הרי שהעלות שלו היא מספר עצום של שעות עבודה עליו ובעיקר... המון שעות של תכנון והמון מחשבות...

חשבנו הרבה ותכננו הרבה על מנת לכתוב "קוד נקי" (קוד שקל לקרוא ולהבין) ולהימנע מ-"קוד ספגטי" (קוד מבולגן ולא איכותי).

כפי שכבר ציינתי - לאורך הדרך, היו לנו הרבה תכנונים ומחשבות לגבי המשך העבודה על הפרויקט, אבל האם שעות המחשבה כה רבות על כל שם של משתנה או פונקציה היו שוות את ההשקעה של הזמן? האם זה כל כך רע שיהיה לנו בהתחלה קוד ספגטי בתוך הפרויקט? האם רפקטורינג בשלב יותר מאוחר - לא היה עוזר להשיג קוד איכותי ונקי יותר? אם לענות במשפט אחד - כן ולא. ובמאמר זה אני אסביר לכם כמה טיעונים בעד או נגד "קוד ספגטי".

קוד ספגטי קיים אצל כולם

אל תדאגו בנוגע לזה שיש (או עלול להיות) אצלכם קוד ספגטי - אתם לא לבד! הנה כמה דוגמאות:

- התאריך הוא 25 בינואר לשנת 2023. לרשת דולפים כ-45GB של קוד מקור של מוצרים שונים מחברת ענק (בתחום ה-IT) הרוסית YANDEX[1].
- מרץ 2023 הייתה דליפת קוד בחברת X (טוויטר של הימים ההם)[2].
- היו גם דליפות מידע מענקיות כמו גוגל ומייקרוסופט, שאגב לא פעם בוצעו ע"י עובדים ממורמרים תוך כדי עזיבה או לאחריה, ואפילו תוך כדי העסקה בחברה.

> "אנחנו יכולים לחבוש תמיד אך ורק "כובע אחד""

אתם בוודאי שואלים - איך זה קשור לנושא שלנו? לפי העדויות של אנשים שנחשפו וקראו אפילו חלקים קטנים של הקוד שדלף, ראו שם לא רק קוד ספגטי, אלא קוד לא איכותי שפשוט מתחת לכל ביקורת!

משהו שאפילו לרגע לא הייתם מצפים לראות בקוד מקור בחברות ענק שבדרך כלל אמורות לתת דגש על קוד נקי וקריא שקל לתחזק אותו ולכן גם דרישות סף למשרות עבודה בחברות כאלה הוא בדרך כלל גבוה יותר מהממוצע בשוק. אבל העובדות מראות תמונה שונה לחלוטין, גם בחברות ענק כותבים קוד לא איכותי בעליל! זאת אומרת זה קוד שנכתב ע"י מהנדסים מצוינים, נבדק ועבר סקר קוד (מה שמכונה code review) של מהנדסים מנוסים ומבריקים יותר ונהיה חלק מפרויקט כזה או אחר שאנחנו משתמשים בו היום, כמו למשל ג'ימייל.

ופה נשאלת השאלה - האם האפשרות שמנוע חיפוש של גוגל כתוב בצורה לא איכותית - הפריעה לכם אי פעם ובגלל זה חדלתם להשתמש במוצר הזה או בכל מוצר אחר של חברת גוגל? מניח שאם כבר אתם מעדיפים מוצר אחר - זה בעיקר בגלל נוחות השימוש (מה שמכונה User Experience) של מוצר אחר, ולא בגלל קוד לא איכותי.

תכתבו קוד באחד משני הכובעים

בחלק הזה - אני רוצה לחזור איתכם חזרה לפרויקט האוטומציה שהוזכר לפני כן. יצרנו משהו מדהים ביותר, שעזר לייעל המון תהליכים בחברה, למרות שכצוות היינו בעצמנו די בתחילת דרכינו כמפתחי אוטומציה. מישהו היה מנוסה יותר, מישהו פחות. מישהו היה באמצע תואר ראשון במדעי המחשב, מישהו היה בוגר קורס אוטומציה. אולי זה מה שעזר לנו בהרבה מאוד מקרים - כי פשוט דהרנו קדימה מהתלהבות שאנחנו מייצרים משהו שעובד יותר ויותר טוב. בהסתכלות אחורה, בעיניים יותר מנוסות, היו אינסוף דברים שאפשר היה לעשות אחרת וטוב יותר. למשל עניין התכנון. זוכרים שציינתי שהפרויקט עלה המון שעות מחשבה ותכנון? כי רצינו לעשות את הדברים כמה שיותר גנריים ולהימנע מקוד ספגטי. כשאני משתמש במושג "גנרי" - אני מתכוון למשל לפונקציות שיודעות לטפל בהמון מקרים, ואז אומנם זה עושה את הפונקציה גדולה יחסית מבחינת כמות שורות קוד, אבל מנגד מאפשר להכניס הרבה לוגיקה לפונקציה אחת. בעצם אחת המטרות העיקריות של קוד גנרי היא לעשות שימוש חוזר באותו קוד (בהרבה מאוד מקרים זה קריאה חוזרת לאותה הפונקציה) במקרים שונים.

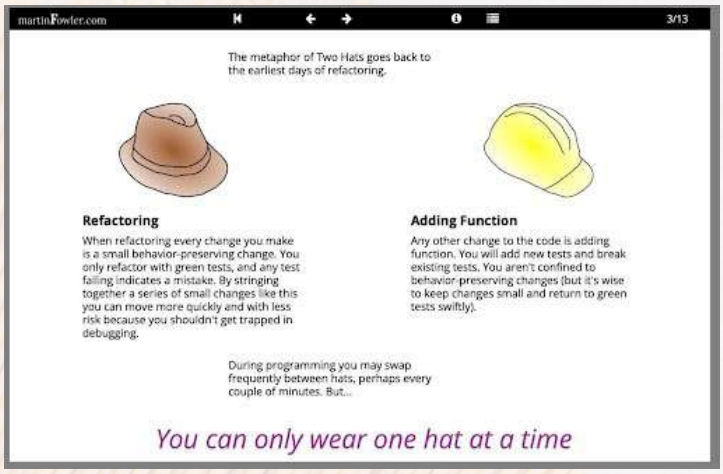
אז המרוץ הזה אחרי משהו גנרי - גרם לנו לכמה שגיאות. העיקריות שביניהן:

- ניסיון לחשוב ולטפל בכל המקרים ותגובות.
- שיפור בקוד (מה שמכונה רפקטורינג) הקיים תוך כדי כתיבת קוד חדש.

אני זוכר שישבתי יום אחד עם אחד המפתחים המנוסים בחברה ודנו קצת על הדיזיין של הפרויקט והוא נתן לי טיפ מאוד מאוד חשוב: **לא לנסות לחשוב על הכל ולתכנן את הכל, כי לא משנה איך נתכנן את הדברים, תמיד יהיו דברים לא צפויים או כאלה שלא לקחנו בחשבון ותמיד יהיו חלקים של שיפורים והתייעלות.**

בכנות - לא האמנתי לו בהתחלה. ניסינו בכל מקרה לחשוב על כל דבר, שיחקנו עם קטעי קוד קטנים שעבדו כי פתאום חשבנו על שם אחר למשתנה או לפונקציה וככה היינו במרוץ אחרי הזנב של עצמנו. אחת הבעיות הנוספות שהייתה לנו - שפשוט המצאנו כללים וחוקים תוך כדי תנועה. אנחנו עוד נחזור לזה באחד הסעיפים הבאים.

ככל שהפרויקט התקדם - התחלנו לראות שאנחנו לא תמיד עומדים ביעדים שלנו, ולא מעט פעמים אפילו משימות קטנות מוגשות באיחור. מרוץ אחר קוד מושלם גרם לעיכובים. ופה בדיוק מגיע החלק של שני כובעים.



אלכס קומנוב

בעל מספר שנים בתחום הבדיקות האוטומטיות. עובד כמפתח אוטומציה ותשתיות אוטומציה בחברת Ibex-ai נשוי+1, חובב נגינה על גיטרה וטיולים ובעל ערוץ יוטיוב לבדיקות אוטומציה

1 https://www.linuxadictos.com/iw/la-filtracion-del-codigo-yandex-revela-varios-detalles-de-clasificacion-del-motor-de-busqueda-ruso.html
2 https://www.pc.co.il/featured/383720/

אל תפחדו מקוד ספגטי בתוך פרויקט אוטומציה אלכס קומנוב

אנחנו יכולים לחבוש תמיד אך ורק "כובע אחד". באיזה כובעים מדובר? כאן זה המקום להסביר:

- כובע אחד - כתיבת קוד חדש. הוספת פיצ'רים, פונקציות, קלאסים וכו'.
- כובע נוסף - שיפור הקוד, מה שמכונה רפקטורינג. שיפור של הקוד הקיים.

הכוונה בציטוט הזה שאנחנו צריכים להפריד את מטרות בכתיבת קוד. אם אנחנו רוצים לממש משהו חדש - אז אנחנו מתעסקים אך ורק בזה. לא פוזלים למשהו אחר, לא משפרים משהו קיים. ברגע שהגענו למצב שאנחנו רוצים/צריכים לשפר משהו (כי הטסטים נשברו, כי זיהינו איזה מחדל שדורש שיפור). במקרה של רפקטורינג - אנחנו מתעסקים נטו ברפקטורינג. כמובן שיכול להיות שתוך כדי כך נכתוב פונקציות חדשות על מנת לשפר/לתקן משהו שנשבר. הכוונה שאנחנו לא מתעסקים במימוש של פיצ'ר חדש תוך כדי הרפקטורינג. אגב, קל מאוד לממש את מטרותיהם של שני הכובעים האלה, למשל ע"י פתיחת משימות (Tasks) בגי'רה (או משהו מקביל לזה). במקרה כזה - אתם תדעו שיש לכם משימה של מימוש ומשימה של רפקטורינג.

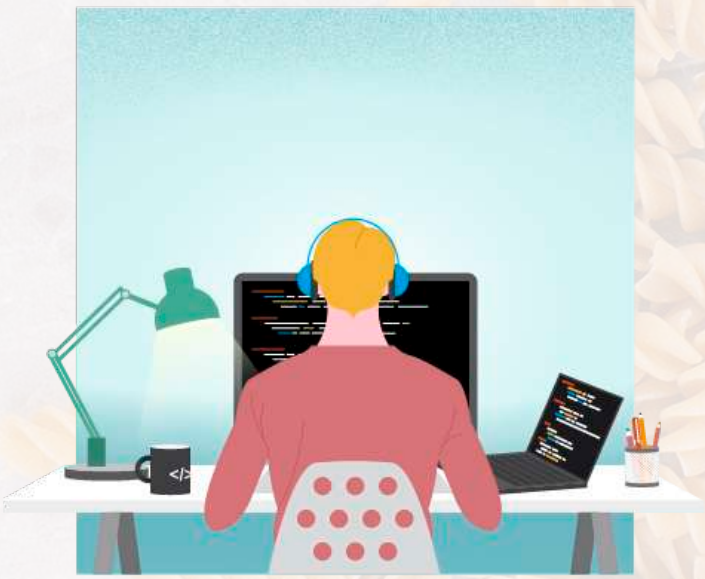
לנו זה מאוד עזר להתמקד, לנצל ולנהל את הזמן בצורה חכמה יותר. והדברים התחילו להתקדם.

דרך אגב, יכול להיות שלא תצטרכו בכלל לחבוש את הכובע של הרפקטורינג על משהו שכתבתם:

- סיבה אחת - כתבתם משהו בצורה מעולה. זה גם עובד וגם עובר כל סוג של בקרה.
- סיבה נוספת - כתבתם משהו שיהיה לזה שימוש ממש מועט. ובמקרה כזה - ככל הנראה לא יהיה בכלל כדאי להשקיע ברפקטורינג של זה. זה עובד. זה עונה על הצרכים - לא בטוח שמקרה כזה צריך יותר מזה.

> "אין הכי טוב – יש את מה שהכי מתאים בשביל הצורך הספציפי שלכם"

אני רוצה לתת פה סייג קטן. זה לא אומר שצריך במכוון לכתוב קוד רע. צריך עדיין לכתוב ולשמור על כל מיני כללים כמו:
עקרון של DRY - ראשי תיבות של Don't Repeat Yourself. העקרון מדבר על זה שצריך להימנע מכתיבה חוזרת של הקוד ובמקום זה לעשות שימוש חוזר בפונקציונליות שכבר נכתבה.
טכניקת תכנות POM - כל עמוד באפליקציה/אתר שבשבילם נכתבת האוטומציה יקבל ייצוג על ידי מחלקה (class) נפרדת. בכל מחלקה נממש את זיהוי האלמנטים שבעמוד ונכתוב את הפונקציות / מתודות רלוונטיות. לאחר מכן בטסטים נקרא ונפעיל את מה שרלוונטי לאותו הקלאס (במייצג עמוד זוכרים?).
ככה אנחנו מפרידים בין עמודים לקלאסים, ככה אנחנו כותבים דברים במקום הייעודי להם. זה מקל על התחזוקה, זה מאפשר שימוש חוזר בקוד שכבר כתוב (זוכרים את DRY? העקרון הקודם?).



היצמדו לצורת כתיבה מסוימת מאוד הכרחית במיוחד אם כבר יש כללים שנבחרו שיהיו חלק מעבודה. צריך כמובן לחלק לפונקציות בצורה נכונה. לעשות פונקציות קטנות יותר. לתת שמות הגיוניים.

רפקטורינג של קוד שיהיה יפה ולא רק עובד במקרה שעוד מעט יש תאריך הגשה סופי של המשימה - לא בהכרח רלוונטי ואפילו יכול להיות מיותר.

קודם כל תגרמו לקוד לעבוד

בהמשך לחלק הקודם - אני רוצה להדגיש את החשיבות של עמידה בזמנים. תתארו סיטואציה שאתם לא הספקתם להגיש משהו כי ניסיתם לגרום לקוד להיראות יפה. אבל שכחתם את הדבר הכי חשוב - שיותר חשוב שהקוד יעבוד. ופה אני רוצה להראות לכם ציטוט של רוברט מרטין (מחבר **הספר** clean code):

"1. "First make it work." You are out of business if it doesn't work. 2. "Then make it right." Refactor the code so that you and others can understand it and evolve it as needs change or are better understood. 3. "Then make it fast." Refactor the code for "needed" performance."

אנחנו צריכים לחלק את העבודה ל-3 חלקים:

1. לגרום לקוד לעבוד.
2. לכתוב את הקוד נכון.
3. לשפר את הביצועים.

אם אנחנו לא נגיש לפחות גרסה ראשונית ועובדת בזמן - אנחנו לא עמדנו במשימה. למי יהיה אכפת שניסיתם לגרום לזה להיראות יפה? לא פעם ניצבנו בסיטואציה שכזו, שחשבנו יותר מידי על ואפילו היו לנו אחלה רעיונות - אבל לא עמדנו במשימה ואפילו משהו ראשוני לא היה להראות. וזה היה קצת מביך אפילו.

לכן קודם כל תתחילו עם משהו ראשוני שיהיה אפשר לראות איך זה עובד. וכן - גם פה אנחנו נוגעים בקוד ספגטי. אל תפחדו לרשום קוד ספגטי שישיג את המטרה העיקרית, שהדברים יעבדו. תכתבו כמה טסטים פשוטים, ,תבדקו שתסריט בסיסי (למשל התחברות למערכת) עובד ועובר בהצלחה. אולי אתם בכלל תתקלו בבעיה כזו או אחרת? למשל שהריצה לא מצליחה (למשל בגלל סיבה טכנית כזו או אחרת).

אחרי שגרמתם לקוד לעבוד - זה הזמן לשפר את הנראות של הקוד. בשלב הזה מומלץ להיצמד לכלל - נגעתם בקוד, צריך להפוך אותו טוב יותר ואיכותי יותר מאשר היה לפני. אחרת, חבל על הזמן שלכם. אם לא שיפרתם שום דבר - אז למה נגעתם בקוד שעובד?

ובסוף - לגרום לזה לעבוד מהר יותר. למשל להוריד את זמני הריצה של הטסטים.

כתיבת קוד זה מסע למידה

בואו נחשוב על סיטואציה שאנחנו כתבנו כמה קטעי קוד, גם אם זה קוד ספגטי. ואז בחלוף זמן מסוים הבנו שיש צורך בשיפור והתחלנו לשפר. יכול להיות שנשפר כי יש באגים אצלינו בפרויקט. יכול להיות שנשפר כי נרצה לממש איזו צורת קוד חדשה (למשל נכניס פיצ'ר חדש של השפה שיכול להיות לנו מאוד שימושי).

זו אחת הדרכים הכי טובות ללמוד לכתוב קוד איכותי. כל הבנה של סיבה כזו או אחרת שמביאה לצורך של לשפר את הקוד - גורמת לנו ללמוד ולהבין את הסיבה לעומק. וכמובן יש פה את החלק של החקירה אחר האמת, שגם כן מהווה לימוד מאוד טוב של Debugging.

יש מקור נוסף ללמידה. תנו לאנשים המנוסים מכם לתת לכם הערות. זה גם דרך מאוד מאוד טובה ללמוד ולחוות מהניסיון של המומחים. זה יכול לבוא כהערות בתוך המשימות, או תוך כדי שיחה על כוס קפה במסדרון. כמובן שצריך להגיד את זה בזהירות. יש פה הבדל מאוד מאוד דק בין הערות שניתן לקבל מאותם המומחים שהן הערות בונות (סתם לדוגמה שימוש חכם יותר במשתנה על מנת לקצר את קוד או הערה על מוסכמות כל שהן שלא ידעתם ומפנים אותכם לדוגמאות של צורת הכתיבה הנדרשת) לבין הערות חוזרות על עצמן על אותם

דברים שכבר קיבלתם הערות לגביהם ואז זה קצת יכול להצביע על זה שאתם צריכים טיפה יותר לשים לב ולא לעשות את אותן טעויות, בעצם ללמוד מהן כמה שיותר מהר ובאופן יעיל.

חוץ מזה - לפעמים רפקטורינג רציני - יכול להביא שוב פעם מסע של למידה. כי אולי תממשו משהו חדש שבעצם ידרוש אדפטציה לצורת כתיבה אחרת. וזה אומר שצריך יהיה ללמד אנשים מחדש ואולי אפילו להאט את קצב התקדמות בפרויקט ולהדוף ביקורת של צוותים אחרים שימחו על זה שעוד פעם יש משהו שצריך ללמוד ושלאף אחד אין כוח לזה. 🙂

היצמדו לכללים הקיימים

אני רוצה לסיים את המאמר שלי עם הכלל שאולי הכי חשוב, היצמדו לכללים שכבר מוסכמים חברה/בצוות. למשל במקרה שיש כבר קונבנציות מסוימות (נותנים שמות לפונקציות ומשתנים בצורה מסוימת), הפרויקט בנוי בצורה מסוימת (מבנה תיקיות) - אין צורך להמציא את הגלגל מחדש. תכתבו כמקובל.

אל תשכחו את הנושא של תחזוקת הקוד. קוד אמור להיות מה שנקרא maintainable, כזה שקל יחסית בבוא המקרה לשנות בצורה קלה ומהירה. במיוחד אם כבר יש איזה שהם תהליכים שמקובלים - זה אומר שרוב המפתחים עובדים בצורה מסוימת וגם מבינים את המבנה ואת הקונבנציות, וקל להם לקרוא את הקוד שנכתב בצורה מסוימת. וככה גם תקבלו את ההערות בקוד שלכם, בהתאם למקובל. זאת אומרת - יבקשו מכם בתהליך של code review להיצמד לאותם הכללים.

אני רוצה להתייחס כאן לעוד נושא שמאוד מאוד שונה בין הצוותים, אפילו באותה חברה, וזה נושא התיעוד. יש ה-מ-ו-ן טכניקות וכלים שמאפשרים לתעד את הקוד החדש או הקיים בצורה מעולה. זה תמיד מאוד סובייקטיבי כי יש כאלו שמאמינים שצריך לתעד בתוך הערות ליד הקוד הכתוב, יש כאלה שמאמינים שצריך לתעד דרך שמות אינפורמטיביים של משתנים ופונקציות ויש כאלה שיגידו שצריך לנהל את כל התיעוד במקום חיצוני לקוד שלכם. גם פה יש עצה אחת - תיצמדו לכללים. תבררו איך זה מתבצע אצלכם בצוות במקרה ומשהו כזה לא קיים - אולי תביאו בעצמם איזה רעיון, תגבשו איזה נוהל של עבודה עם התיעוד. אגב, בדרך כלל מה שבדרך כלל משותף בנוגע לתיעוד של הקוד, זה שאף אחד לא אוהב לתעד. 🙂

במקום סיכום

להיות מפתח - זה קצת מורכב בימינו, בין אם זה בצוותי פיתוח ובין אם זה בצוותי האוטומציה. לכן אל תמהרו רק לכתוב וזהו ולשגר את המשימה. תחשבו קצת, תבדקו מה כבר מקובל מבחינת תהליכי עבודה קיימים. תשקיעו בתהליך ה-onboarding אם אתם נכנסים לחברה חדשה או לצוות חדש. תשאלו שאלות (ולא רק את המומחים). יש המון כללים, קונבנציות, קונספציות, טכנולוגיות שאפשר להיעזר בהם. אפשר לפגוש המון שאלות ברשתות כמו "מה הכי טוב?". התשובה שלי - אין הכי טוב - יש את מה שהכי מתאים בשביל הצורך הספציפי שלכם.

נותני החסות

PractiTest
cloudbeat
PROVENGO
spiraTest
JOHN BRYCE
ITCB Israeli Testing Certification Board
inflectra
Testing & Automation GeekWeek
North High Tech Park

ISTC 2024
תחרות גביע הבדיקות 2024
מתחילה

- סבב ראשון של שלב המוקדמות של התחרות יתקיים באופן פרונטלי ביום רביעי 08 במאי 2024 בין השעות 17:30-21:00 בפארק הייטק צפון (בר-לב).
- סבב שני של שלב המוקדמות של התחרות יתקיים באופן מקוון ביום שישי 10 במאי 2024 בין השעות 08:30-12:00.
- סבב שלישי של שלב המוקדמות של התחרות יתקיים באופן מקוון ביום רביעי 15 במאי 2024 בין השעות 17:30-21:00.
- הצוותים הטובים ביותר יעלו לגמר הגדול שיתקיים בכנס Testing&Automation GeekWeek 2024 ביום ראשון 23 ביוני 2024.

מקום ראשון
יומיים חינם בכנס AgileTesting Days שיתקיים בגרמניה, נובמבר 2024 וכולל טיסה ולינה בחסות משותפת עם ITCB

מקום שני
קורס מקצועי + אוזניות Sony איכותיות בחסות משותפת עם ג'ון ברייס וכלל נותני החסות

מקום שלישי
קורס מקצועי בחסות ג'ון ברייס

www.istc.org.il

מה הסקרים אומרים?...

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

טור זה מציג ממצאי סקרים המציגים מגמות במרחבי העולם לעיונכם.
בגיליון זה אנו מציגים חלק מתוצאות הסקרים של סיכום שנת 2023

תוצאות הסקרים נלקחו מ-Katalon.com

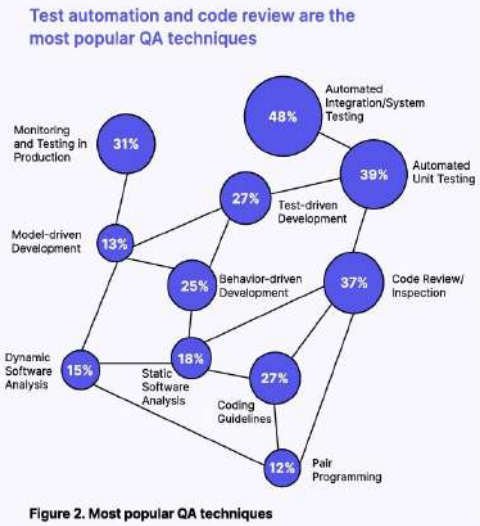


Figure 2. Most popular QA techniques

כדי להשיג יעדי QA צוותי התוכנה יכולים להשתמש במספר טכניקות בדיקה החל מסקירת קוד והנחיות קידוד ועד לבדיקות תוכנה. האיור הנ"ל מתאר את טכניקות הבדיקה המיוצגות באמצעות עיגולים, כאשר המספר מציין את אחוז המשיבים שאמרו שהם משתמשים בטכניקה. הקשר בין שתי הטכניקות מצביע על כך שהנחקרים נוטים להשתמש בהן יחד. מלבד בדיקות ידניות, ארבע טכניקות הבדיקה הנפוצות ביותר הן אינטגרציה/בדיקות מערכת אוטומטיות (48%), בדיקות יחידות אוטומטיות (39%), סקירת קוד (37%), וניטור ובדיקה בייצור (31%), כלומר בדומה לתוצאות הסקר של השנה שעברה. כ-82% מהמשיבים אמרו שהצוותים שלהם יישמו לפחות אחת מארבע טכניקות הבדיקה הללו בפרויקטים שלהם. TDD (27%), הנחיות קידוד (27%) ו-BDD (25%) הם גם טכניקות נפוצות בשימוש על ידי צוותים. צוותי תוכנה משתמשים לעתים קרובות באוסף של טכניקות בדיקה הקשורות יחד כדי להשלים זה את זה. צוות שמפעיל בדיקות יחידות אוטומטיות נוטה לבצע גם אינטגרציה ובדיקות מערכות אוטומטיות, כפי שמוצג על ידי קו החיבור בין הטכניקות הללו. צוותים עם בדיקות יחידות אוטומטיות מתרגלים כנראה TDD וסקירת קוד ביחד. באותו אופן, צוותים נוטים ליישם סקירת קוד, בדיקות יחידות אוטומטיות, הנחיות קידוד, ניתוח תוכנה סטטי ותכנות זוגיות יחד.

Figure 3. Most effective QA techniques

טכניקות הבדיקה הנפוצות ביותר שדווחו על ידי המשיבים הן גם היעילות ביותר. כ-68%, 67% ו-57% מהמשיבים שיישמו, בהתאמה, אינטגרציה ובדיקות מערכות אוטומטיות, בדיקות יחידות אוטומטיות, סקירת קוד ובדיקת קוד הסכימו כי טכניקות הבדיקה אלו הן היעילות ביותר. פיתוח מונע מודל (42%), תכנות זוגי (40%) |37%) (TDD), לעומת זאת, נתפסים כיעילים במידה מסוימת. רק שליש מהמשיבים שיישמו ניטור ובדיקות בייצור, ניתוח תוכנה דינמי, ניתוח תוכנה סטטי BDD אמרו כי הטכניקות הללו היו היעילות ביותר. אינטגרציה/בדיקות מערכות אוטומטיות, בדיקות יחידות אוטומטיות ובדיקת/בדיקת קוד נחשבות רבות כטכניקות בדיקה יעילות מכיוון שהן יכולות לזהות ולטפל בבעיות בתוכנה בשלב מוקדם בתהליך הפיתוח, מה שיכול לחסוך זמן ומשאבים בטווח הארוך. TDD, ניטור ובדיקה בייצור, BDD וקידוד נחשבים גם הם יעילים מכיוון שהם מסייעים להבטיח שהתוכנה עומדת במפרטים הנדרשים ואיכותית. ניתוח תוכנה דינמי וניתוח תוכנה סטטי הם הטכניקות הטובות ביותר לזיהוי בעיות פוטנציאליות בקוד תוכנה. פיתוח מונחי דגמים ותכנות צמד הם טכניקות חזקות לתכנון ופיתוח תוכנה באיכות גבוהה.

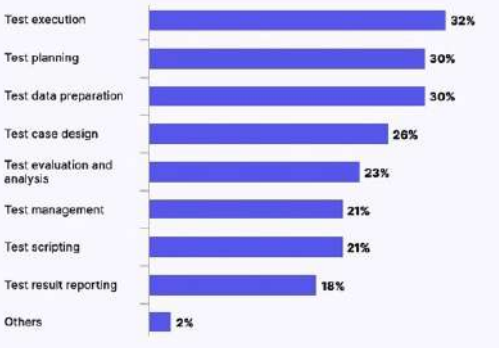


Figure 4. The most time-consuming testing activities

צוותי בדיקה צריכים להבין היכן עליהם להתמקד בכל הנוגע לשיפור הפרודוקטיביות. סקר זה מראה כי שליש מהנשאלים אמרו שהפעילויות אשר גוזלות את רוב זמנם הן ביצוע בדיקות, תכנון בדיקות והכנת נתוני בדיקה – תלוי איזה סוג בדיקה מתבצעת. בדיקות ידניות אשר דורשות פרק זמן משמעותי לביצוע, היא מסורבלת עבור מהנדסי בדיקות המוטלים על ביצוע בדיקות נסיגה (Regression) במספר סביבות. מהנדסי בדיקות אוטומציה מבלים את רוב זמנם בתכנון בדיקות והכנת הנתונים. בעוד ששליש מהם מדיווחו שביצוע בדיקות הינה המשימה אשר גוזלת את רוב הזמן שלהם, מהנדסי בדיקות אוטומציה דיווחו שיש להם יותר זמן להיבטים אחרים של עבודתם. דיווח תוצאות הבדיקה אינו מזוהה כפעילות הגוזלת ביותר על ידי רוב המשיבים. תוצאות אלו מצביעות על כך שכמות משמעותית של זמן מושקעת בביצוע בדיקות והכנת נתוני בדיקה במהלך תהליך הבדיקה, בעוד שפעילויות אחרות כגון ניהול בדיקות ודיווח תוצאות הבדיקה תופסות חלק קטן יותר מהזמן.

מה הסקרים אומרים?...

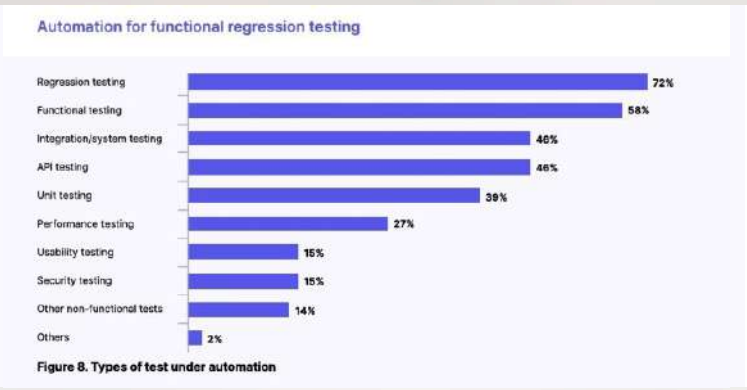


Figure 8. Types of test under automation

אוטומציה משמשת את צוותי התוכנה למגוון פעילויות בדיקה וסוגי בדיקה. ברור שרוב הנשאלים (72%) השתמשו באוטומציה לצורך בדיקות נסיגה (Regression). כאשר צוותים צריכים לאמת במהירות פונקציונליות שהם כבר בדקו במחזורים אחרים, זה סוג הבדיקה שעבורו האוטומציה מתאימה ביותר. ביצוע בדיקות פונקציונליות (58%), אינטגרציה/מערכת (46%) ו-(46%) API הם גם סוגים פופולריים של בדיקות המאמצות אוטומציה.

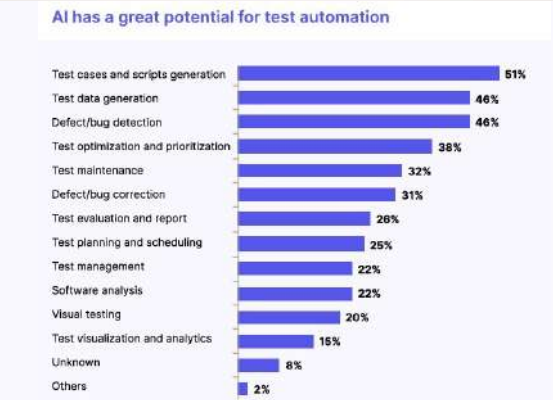


Figure 19. Which of the following activities would you want automation tools to be more intelligent and autonomous?

כשנשאלו לאילו פעילויות בדיקה הם ציפו שכלי אוטומציה של בדיקה יהיו חכמים ואוטונומיים יותר, המשיבים בחרו ביצירת מקרה/סקריפטים של בדיקה, יצירת נתוני בדיקה, זיהוי פגמים ואופטימיזציה של בדיקות כארבעת העדיפויות העליונות. אלו הן פעילויות קריטיות אך גוזלות זמן באוטומציה של בדיקות. תחומים חשובים נוספים שזכו לדירוג גבוה על ידי המשיבים כללו תחזוקת בדיקות ותיקון ליקויים. אלו הם התחומים שבהם בינה מלאכותית תהיה הישימה והמשפיעה ביותר בעתיד. מקרי בדיקה אוטומטיים/יצירת סקריפטים, יצירת נתוני בדיקה, זיהוי באגים ואופטימיזציה של בדיקות הן הפעילויות שעבורן המשיבים רצו להיות חכמים ואוטונומיים יותר באמצעות תמיכה ביכולות AI. בנוסף, תחזוקת בדיקות ותיקון ליקויים הם תחומים שדורגו גבוה על ידי המשיבים בסקר זה.

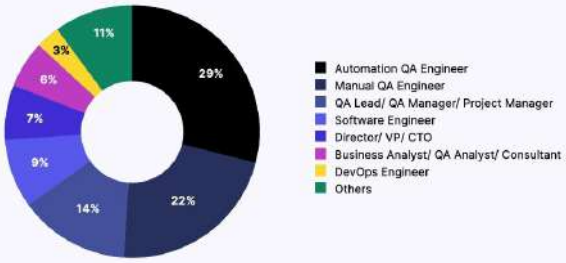


Figure 23. Professional roles of survey participants

הקבוצות הגדולות ביותר של המשיבים הן מהנדסי בדיקות אוטומציה ומהנדסי בדיקות ידניות, המהווים 28% ו-23% מהמשיבים, בהתאמה. תפקידים נוספים שמילאו המשיבים כוללים מוביל/מנהל בדיקות, מנהל פרויקטים (14%), מהנדס תוכנה (9%), תפקידי ניהול בכירים דירקטור/סמנכ"ל / CTO (7%), אנליסט עסקי/אנליסט /יועץ בדיקות (6%), מהנדס DevOps (3%) ואחרים (11%).

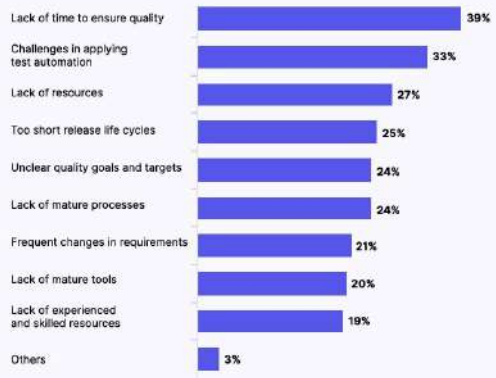


Figure 20. What are your top challenges in achieving your software quality goals?

בסקר, התבקשו מהמשיבים לספר מה מונע מהם להגיע ליעדי האיכות שלהם. התוצאות מראות ששני האתגרים המובילים בהשגת יעדי איכות תוכנה הם המחסור בזמן להבטיח איכות (39%), ואתגרים ביישום אוטומציה של בדיקות (33%). אתגרים מובילים נוספים כוללים היעדר משאבים (27%), מחזורי חיים של שחרור קצר (25%), יעדי איכות ויעדים לא ברורים (24%) והיעדר תהליכים בשלים (24%).

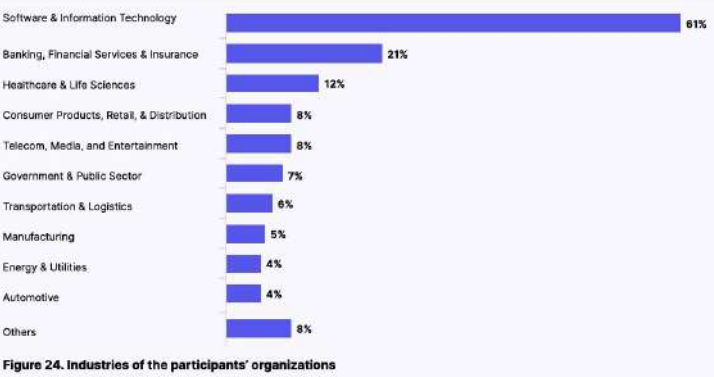


Figure 24. Industries of the participants' organizations

רוב משתתפי הסקר עבדו במגזר התוכנה וטכנולוגיית המידע (61%), אך הם עסקו בפיתוח תוכנה ובפעילות הנדסת איכות למגזרים אחרים. הסקר כולל גם משתתפים שארגונים פעלו בתחום הבנקאות, השירותים הפיננסיים והביטוחים (21%); בריאות ומדעי החיים (12%); מוצרי צריכה, קמעונאות והפצה (8%); טלקום, מדיה ובידור (7%); ותעשיות אחרות.

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®

זמנים 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 #14 – שורטקאסט "בדיקות חוקרות"

בואו לשמוע את קובי יונסי מרצה מוביל בתחום הבדיקות מציג את אחת הטכניקות בדיקה מובנות נסיון "בדיקות חוקרות - Exploratory Testing על פי הסמכת ISTQB - איך מבצעים בדיקות חוקרות?, איך מתחילים?, מה החוקים? ועוד.

פרק #15 – ראיון עם אלון פרידמן ויסברד – בדיקות נגישות (המשך)

חלק ב' בנושא נגישות עם אלון פרידמן ויסברד, מומחה נגישות דיגיטלית בחברת Wix. בפרק זה אנו נתמקד בכלים ובאיך בודקים נגישות. נצלול לבדיקות הנגישות עצמן. נכיר כלים ונעשה את הבדיקות hands-on, כאשר למעשה מדובר בפורמט של פודקאסט, המבוסס על שמיעה, אנחנו נתאר הכל באופן שיהיה נגיש למאזינים וגם לעיוורים

פרק #16 – ראיון עם מובילת בדיקות – קארין זלוף

בואו לשמוע את קארין מספרת על הדרך שעשתה ב-10 שנים האחרונות בתחום הבדיקות, על היוזמות שהניעה, האתגרים ועל קפיצות ההתפתחות ששאפה אליהן בשתי החברות בהן עבדה.

פרק #17 – ראיון עם מוביל בדיקות נחום דימר מחברת Cloudbeat

בואו לשמוע את נחום מספר על הדרך שעשה ועל חייו האישיים עד שהגיע לתחום הבדיקות, איך עבר לעולם היזמות ולמה הקים דווקא חברת מוצר אוטומציה.

פרק #18 – בדיקות מבוססות מודל עם ד"ר מיכאל בר-סיני

בואו לשמוע את נתנאל הרוש וד"ר מיכאל בר-סיני מחברת Provengo מדברים על בדיקות מבוססות מודל.

פרק #19 – "ראיון המעלית" עם לימור שחר, טיפים למחפשי עבודה

בואו להקשיב ללימור שחר משנה למנכ"ל וסמנכ"לית משאבי אנוש בקבוצת UCL מסבירה על כל מה שצריך לדעת כאשר מחפשים עבודה בבדיקות תוכנה

פרק #20 – שורטקאסט "טבלת החלטות"

בואו לשמוע את קובי יונסי מרצה מוביל בתחום הבדיקות מציג את אחת הטכניקות בדיקה קופסא שחורה "טבלת החלטות" – (Decision table) על פי הסמכת ISTQB – איך מתכננים מינימום מקרי בדיקה למקסימום כיסוי דרישה.

קישור לערוץ הפודקאסט לנו bit.ly/TestIL_Podcasts

בודקים FUN בכיף

Agile Testing – תפזורת מושגים

I	B	O	U	N	D	A	R	Y	V	A	L	U	E	A	N	A	L	Y	S	I	S	Q
O	T	Y	N	F	X	Q	G	N	I	T	S	E	T	A	T	E	B	R	J	H	V	Q
D	O	T	S	E	T	N	O	I	T	A	C	I	F	I	R	E	V	D	L	I	U	B
D	X	I	N	T	N	E	M	E	G	A	N	A	M	T	C	E	F	E	D	R	Y	N
Q	H	L	U	N	J	Q	T	Y	T	I	X	E	L	P	M	O	C	C	B	I	R	B
G	P	I	Z	H	E	E	Q	G	N	I	S	S	E	U	G	R	O	R	R	E	S	G
N	G	B	L	A	C	K	B	O	X	T	E	S	T	I	N	G	Q	N	Q	D	X	U
H	N	A	O	T	S	E	F	I	N	A	M	E	L	I	G	A	M	D	K	S	A	I
R	I	L	M	L	C	J	F	W	J	O	L	I	X	W	L	G	N	H	O	H	C	U
Y	G	I	R	A	I	H	Y	N	W	B	C	O	V	E	R	A	G	E	M	Y	C	O
F	G	A	G	G	N	I	T	S	E	T	A	H	P	L	A	V	E	M	N	C	U	L
X	U	V	D	T	A	A	U	D	I	T	U	X	F	R	R	W	E	V	U	H	R	H
L	B	A	A	I	R	E	T	I	R	C	T	I	X	E	R	R	H	C	O	J	A	O
U	E	E	A	C	C	E	P	T	A	N	C	E	C	R	I	T	E	R	I	A	C	Z
X	D	B	N	A	I	R	E	T	I	R	C	Y	R	T	N	E	N	S	E	G	Y	I
D	I	Y	E	X	P	E	C	T	E	D	R	E	S	U	L	T	X	V	O	K	Y	E

BUILD VERIFICATION TEST
BOUNDARY VALUE ANALYSIS
ACCEPTANCE CRITERIA
DEFECT MANAGEMENT
BLACK-BOX TESTING
EXPECTED RESULT
AGILE MANIFESTO
ENTRY CRITERIA
ERROR GUESSING
ALPHA TESTING
AVAILABILITY
EXIT CRITERIA
BETA TESTING
COMPLEXITY
DEBUGGING
ACCURACY
COVERAGE
AUDIT

שם הזוכה יפורסם בגליון מס 37
יש לשלוח את הפתרון למייל
MAGAZINE@TESTINGWORLD.CO.IL

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

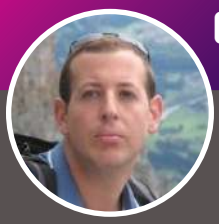
עוסקת בבדיקות תוכנה קרוב ל-20 שנה. עבדה במספר ארגונים בתפקידי בדיקות וניהול בדיקות. בשנים האחרונות עוסקת בפיתוח והוראת קורסים בבדיקות תוכנה ונושאים נוספים

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.

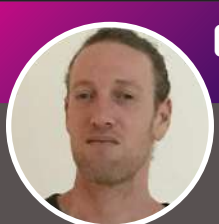


עמית ורטהיימר

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.

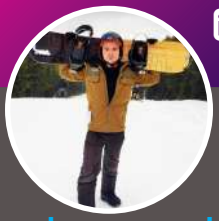


דור רוזנברג

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

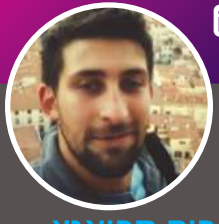
שירה נוסבוים

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