

רבעון שלישי 2019 גיליון מס' 18

מגזין

עולם הבדיקות

www.testingworld.co.il

Airtable

כלי לניהול פרויקטים
ניצן גולדנברג

הקמת מעבדת ביצועים
בחברת SentinelOne
תומר אלון

7 טיפים
לבדיקות WEB
עוז צופי

The big bug hunt
ניצן גולדנברג

מבדקים אוטומטיים בקנה
מידה גדול
אורי קופרשמיד

תוכן העניינים

- 3..... [Airtable](#) כלי לניהול פרויקטים | [ניצן גולדנברג](#)
- 6..... מחפש צרות – כלבים וחתולים | [מיכאל שטאל](#)
- 7..... בחן את עצמך - שאלה | [טל פאר](#)
- 8..... הקמת מעבדת ביצועים בחברת SentinelOne
[תומר אלון](#)
- 10..... בחן את עצמך - תשובה | [טל פאר](#)
- 11..... 7 טיפים לבדיקות WEB | [עוז צופי](#)
- 13..... The big bug hunt | [ניצן גולדנברג](#)
- 15..... אנציקלופדיה לבדיקות - למה צריך בודקים? | [אייל זילברמן](#)
- 16..... מבדקים אוטומטיים בקנה מידה גדול | [אורי קופפרשמיד](#)
- 19..... פינת הטיפ - אל תחזור על עצמך | [קובי הלפרין](#)
- 20..... תחרות הבדיקות הישראלית 2019
- 22..... נגיסה מ-TestIL - מפגשים לבודקי תוכנה | [ניצן גולדנברג](#)
- 24..... מה הסקרים אומרים | [יאן ברון](#)
- 25..... עורכי הגיליון

מו"ל
Israeli Testing Certification Board
ITCB®

ניהול המגזין
iMDsoft, יאן ברון

ניהול התוכן
קובי הלפרין, Nokia

עורכת
איריס קוטליצקי, Testing World

עיצוב גרפי
בית גלי 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

עולם הבדיקות נכתב ע"י בודקים
עבור בודקים
ITCB® מקדמים את קהילת הבודקים
בישראל



www.itcb.org.il



מגזין עולם הבדיקות

עולם הבדיקות הינו מגזין רבעוני. כל הזכויות שמורות. זכויות היוצרים על חומר שפורסם על ידי המפרסם הינן רכושן 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 info.testingworld@gmail.com



Name	Category	Complete	Project photos	Client
1 Tea Packaging	Brand Identity			Bice
2 MOMA Brand Identity	Brand Identity			Mus
3 Codecademy Brand Identity	Brand Identity			Code
4 Mohawk Brand Identity	Brand Identity			Moh
5 Second Home Brand Identity	Brand Identity			Seco
6 NYC Parks Brand Identity	Brand Identity			New

שלום לכולם, בדרך כלל אני לא כותב על כלים אבל הפעם החלטתי לחרוג ממנהגי ולכתוב על כלי שהתחלתי לעבוד איתו בשם **Airtable**.

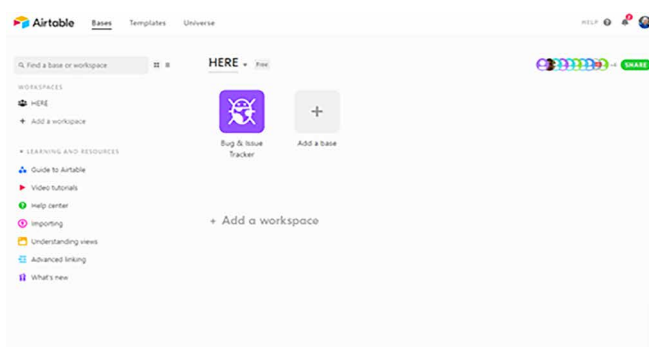
Airtable הוא כלי לניהול פרויקטים שקיים בשוק משנת 2012 העובד בשיטה של בסיס נתונים וגליון אלקטרוני אשר עובד על פלטפורמת ענן.

עיצוב המסכים דומה לגליון אלקטרוני (אקסל) עם אפשרות של צ'ק בוקס, הוספת מס' טלפון, רשימה נפתחת ואפשרות הוספת קבצים. ניתן לעצב את העמודות לפי מה שרוצים או את התכנים שיהיו ברשומות נפתחות.

אז מה יתרונות הכלי הזה ומה הכלי יכול לספק לנו המשתמשים?

1. **מסך ניהול פשוט ונוח**, ניתן לראות את כל הפיצ'רים שאיתם אנו עובדים בצורה נוחה וקלה. ניתן לעצב את המסך על ידי הוספת פיצ'רים חדשים או סידור הפיצ'רים הקיימים בסדר שנרצה. שדה החיפוש נוח, ניתן לראות את כל המשתמשים שרשומים למערכת שלנו, קישורים לסרטוני הדרכה או הסברים נוספים.

דבר נוסף שעשוי להיות מאוד חשוב למי שעובד על כמה פרויקטים במקביל זה האפשרות לפתוח חלל עבודה (Workspace) לכל פרויקט ובתוך כל חלל עבודה לקשר את הפיצ'רים המתאימים לפרויקט.



2. **Project Tracker** - זהו פיצ'ר שעוזר לנהל את משימות הפרויקט. ניתן לנהל משימות קטנות שצריכות רק ניהול זמנים או פרויקטים גדולים שמצריכים שיתופי פעולה עם כמה מחלקות ועובדים בכמה משימות ותת משימות ואפילו ניהול תקציב של פרויקט.

על ידי איחוד כל העבודה במקום אחד ניתן לעקוב אחר ניהול הפרויקט, לשמור על מיקוד של כל חבר צוות ועל מצב המשימות שלוה ועדכון המנהלים לגבי מצב הפרויקט הכולל ציר פרויקט. ניתן לצרף צילומי מסך וקבצי אפיון, לסמן משימות שהסתיימו ולקטלג את המשימות לפי קטגוריות.



ניצן גולדנברג

מזה 5 שנים בתחום בדיקות התוכנה. מהנדס בדיקות בחברת **SeatGeek**, המוביל הראשי של קבוצת המיטאפ **TestIL**, עורך וכתב במגזין "עולם הבדיקות", מרצה בקורסים לבודקי תוכנה וחבר צוות המייצג של **AB** של **ITCB**.

3. **Bug Tracker** - מדובר בפיצ'ר שהכי מדבר אלינו הבודקים "מערכת ניהול באגים". הפיצ'ר מיועד לדיווח וניהול הבאגים ברמה הכי מפורטת שאפשר. ניתן לעצב את הפיצ'ר שיתאים לצרכי המשתמש, בין אם רוצים לסדר את העמודות בסדר מסוים שיהיה הכי נוח לעבודה או להכניס תכנים ברשימה נפתחת, לתייג מפתח מסוים לבאג שבדיוק פתחת או לצרף קבצים לשחזור הבאג. כמובן כל באג מקבל מספר ייחודי משלו.

ID	Status	Created Date	Updated Date	Assignee	Reporter	Comments
1	OPEN	2018-01-01	2018-01-01	John Doe	Jane Doe	There is a bug in the app...
2	IN PROGRESS	2018-01-02	2018-01-02	John Doe	Jane Doe	There is a bug in the app...
3	RESOLVED	2018-01-03	2018-01-03	John Doe	Jane Doe	There is a bug in the app...

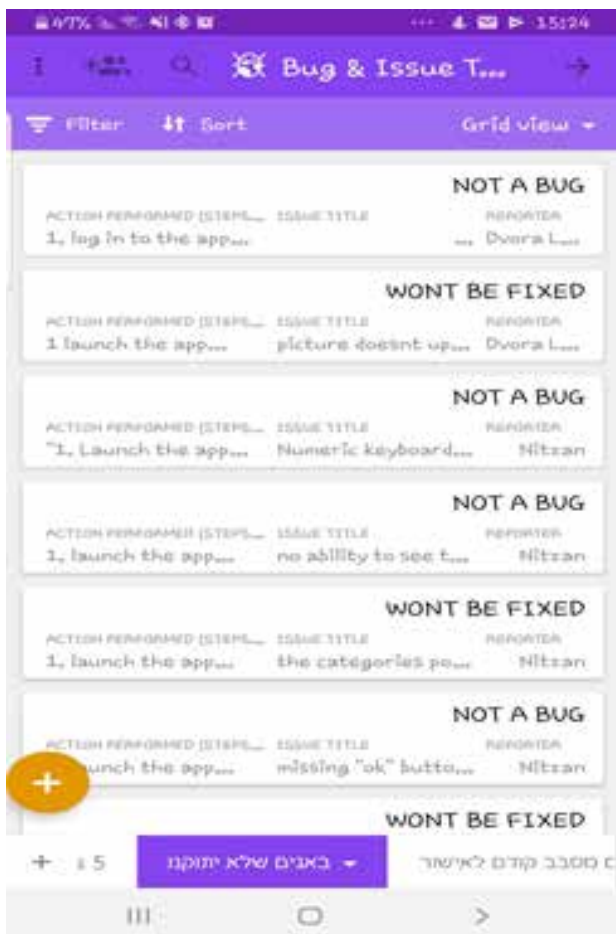
הממשק של Airtable מאפשר לבנות פילטרים אישיים לכל משתמש כך שניתן יהיה להציג את הבאגים לפי קטגוריות שונות, למשל הצגה של כל הבאגים שנפתחו ללא סינון של מצב הבאג או רק את הבאגים הפתוחים או אפילו תצוגה של **Kanban View** וכמובן להעביר כל באג בין עמודות הסטטוס.

OPEN	IN PROGRESS	RESOLVED
There is a bug in the app...	There is a bug in the app...	There is a bug in the app...
There is a bug in the app...	There is a bug in the app...	There is a bug in the app...



ניידים

AirTable מגיע גם עם יישומון לניידים (אנדרואיד ואייפון).



אפשר לייצר תצוגה של כל חברי הצוות ולראות את סך הבאגים שכל אחדות פתח ואת המצב של כל באג.

Name	ID	Status	Assignee
John	1	Open	John
John	2	In Progress	John
John	3	Open	John
John	4	Open	John
John	5	Open	John
John	6	Open	John
John	7	Open	John
John	8	Open	John
John	9	Open	John
John	10	Open	John
John	11	Open	John
John	12	Open	John
John	13	Open	John
John	14	Open	John
John	15	Open	John

במידה ויש באגים שלא יטופלו ניתן להכניס לעמודה ייחודית משלחם.

Name	ID	Status	Assignee
John	1	Open	John
John	2	In Progress	John
John	3	Open	John
John	4	Open	John
John	5	Open	John
John	6	Open	John
John	7	Open	John
John	8	Open	John
John	9	Open	John
John	10	Open	John
John	11	Open	John
John	12	Open	John
John	13	Open	John
John	14	Open	John
John	15	Open	John

לדעתי אחד היתרונות העיקריים של הכלי זו האפשרות לבנות מסמך דיווח באגים שלא מצריך את המשתמשים אפילו להכנס למערכת אלא על ידי מילוי מסמך מקוון אשר שליחתו מכניסה את הדיווח לרשימת הבאגים באופן אוטומטי.

Issue Submission Form

This is a report issue form

Bug/Issue Title *

Please provide a "headline" for the issue that concisely summarizes the problem and why it's important

Issue Type *

How important is it that we fix this issue?

Frequency

Severity

Action Performed (Steps 1,2,3) *

Include more detailed information about the issue, including but not limited to: steps to reproduce the bug, any relevant details needed to help identify and resolve the problem, and more.

Expected Result

Actual Result

Error Message

העבודה עם יישומון נוחה ומאפשרת למשתמש לנהל את המשימות והבאגים גם כאשר הוא לא ליד המחשב. ניתן לראות את כל הלשוניות של קטגוריות הבאגים או לפתוח באג חדש, וניתן לנהל את זמני המשימות.

אז כמה התענוג הזה עולה לנו?

יש 4 מסלולים.

- **המנוי החינמי** (לחלוטין) כולל אפשרות להוסיף בסיסים (פיצ'רים) ללא הגבלה, עד 1200 רשומות (שורות) לכל בסיס, עד 2 ג'יגה ביט של מקום אחסון עם אפשרות שחזור בסיס נתונים עד לשבועיים אחורה.
- **Plus** - עלות של \$10 לחודש בתשלום שנתי או \$12 לחודש בתשלום חודשי כולל 5000 רשומות (שורות) לבסיס, 5 ג'יגה ביט של מקום אחסון עם אפשרות שחזור בסיס נתונים עד ל-6 חודשים אחורה.
- **Pro** - כולל 50,000 רשומות (שורות), 20 ג'יגה ביט של מקום אחסון ואפשרות שחזור של בסיס הנתונים עד לשנה אחורה.
- **Enterprise** - מתאים לחברות שעובדות על מספר רב של פרויקטים ומספר רב של עובדים ונתונים. כולל אי הגבלה של מספר רשומות (שורות) 1 טרה בייט של מקום אחסון עם אפשרות שחזור בסיס נתונים עד ל-3 שנים.



קישור לתמחור המערכת

היתרון במערכת זה שאנו הבעלים של המערכת (גם במסלול החינמי) יכולים להרוויח כסף על כל משתמש שמתחבר אלינו למערכת. על כל משתמש שמתחבר למערכת שלנו אנו מקבלים \$10 קרדיט ועוד \$2 כאשר אותו משתמש הוריד את היישומון.

ניתן להשתמש בסכום שהרווחנו לשדרוג במסלול שלנו.

★ CREDITS

You have **\$122** in credit.

To apply credit to a workspace, go to your workspace's settings page from the left sidebar. ⓘ

Get \$10 in credit for every person you invite.

You've been awarded credit for inviting 12 users

✓ You received \$2 in credit for installing the mobile app.

תמיכה

באתר יש אין ספור סרטונים והדרכות על שימוש במערכת בכל הפיצ'רים. בנוסף יש בלוג אשר ניתן להעלות שאלות, כמו כן ניתן ליצור קשר עם אנשי המכירות של המערכת שיכולים לתת מענה לכל שאלה או בעיה טכנית.

לסיכום

מדובר במערכת ניהול פרויקטים שמתאימה לכל גודל של חברה החל מעובד אחד ועד חברה גלובלית, בין אם אנו עובדים על פרויקט אחד או מספר רב של פרויקטים. ניתן לעצב את המערכת לפי ראות עינינו, ניתן להרוויח כסף על השימוש במערכת. המערכת לא יקרה לעומת מערכות אחרות שאני מכיר ומאוד ידידותית למשתמש. ניתן לעבוד איתה עם דפדפן וואבי או עם הטלפון החכם שלנו. המערכת מתממשקת עם Automate.io, Workato, Asana, Zapier ועוד מלא אתרים ואפליקציות.

TestIL - מפגשים של בודקים למען הבודקים

לקידום מפגשי בודקים - אנו מחפשים

מקום!

מעוניין לארח מפגש בודקים?
הזדמנות להכיר בודקים אחרים ולעודד בקבוצתך התעניינות בבדיקות!

מרצים!

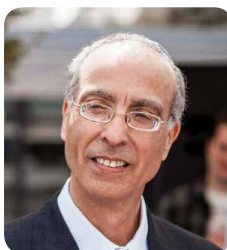


יש לך רעיון או נושא שתרצה לחלוק ולהציג?
הזדמנות מצוינת להצטרף ולהתפתח
בקבוצת המרצים!

test.il@yahoo.com ✉

052-6718757 📞

ניצן 👤



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 כיף).
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

לא מחייב התארגנות גדולה, כל אחד מאיתנו יכול לעשות את זה בלי הרבה טקס מסביב. זה רק דורש מוכנות לוותר על הסיפוק שהערה עוקצנית נותנת ולהיות מוכנים להשקיע משאבי זמן ומחשבה כדי לתת הערות מובנות שניתן לפעול לתיקון.

יוצא לי לקרוא לא מעט מסמכים במסגרת העבודה ורבים מהם קשים להבנה לא בגלל התוכן הטכני אלא בגלל הדרך שבה הם כתובים. ניסוחים מסורבלים, משפטים שאפשר להבין במספר אופנים, חוסר קונסיסטנטיות בדרך שבה קוראים לעצמים ועוד ועוד. אפשרות אחת היא להתעלם ולהאנח: זה מה יש ועם זה ננצח. אפשרות אחרת היא לכתוב הערה קריפטית: "לא מובן". האפשרות השלישית, ובה אני משתדל לנקוט, היא להסביר מה מפריע לי, ולעיתים אף להציע ניסוח חדש. זה לוקח הרבה יותר זמן. קודם כל כי זה אומר שאני כותב משמעותית יותר הערות: לא רק הערות תוכן, אלא גם הערות עריכה. אני מסביר איך הניסוח הנוכחי ניתן להבנה בכמה דרכים או לא ניתן להבנה כלל כי חסרה אינפורמציה. אם הטקסט סותר משהו שכתוב במקום אחר במסמך, אני מפנה לשם או מצטט. וכשצריך להציע

איך לכתוב משהו מסובך באופן ברור וחד משמעי אני מנסה לכתוב הצעה לניסוח חדש, מוחק ומנסח שוב עד שאני מרוצה. המטרה שלי בכל זה היא לא רק להתלונן אלא גם לעזור לצד השני להשתפר. לפעמים התוצאה היא שהכותבים מתקנים ומשפרים את המסמך. לפעמים הם מתעלמים מההערות ופה ושם מישוהו אפילו אומר תודה... במקרה ההפוך, כשאני מקבל מסמך שכתוב היטב, אני דואג לציין זאת לשבח לא רק בפני הכותב אלא גם מיידע את המנהל שלו.

חשוב להדגיש שצריך להיות מאוד ספציפיים על מנת שהמשוב יהיה אפקטיבי. למעשה, ייתכן שאדם בעל אותו רקע של מקבל המשוב לא יבחין כלל בבעיות עליהן נצביע במשוב. הדברים יראו לנו מובנים מאליהם. במשוב, אנחנו צריכים להביא את המקבל לראות דברים בעיניים שלנו. מי שכתבו את הדוח של הבאג או את המסמך שאני קורא ומתקשה להבין, יודעים יותר ממני על הנושא ולכן פרטים שאני מבקש שיוסיפו נראים להם מובנים באופן טריוויאלי. בקשה לעוד מידע שאינה מלווה בהסבר למה הוא דרוש ולא יזהה אי הבנה מגיעים כשהוא חסר, נראית להם כהטרדה, מסמנת אותי כמי שלא עשה שיעורי בית או כסתם אחד שלא ברמה. כאשר ההערה מלווה בהסבר מתאים, יש יותר סיכוי שהכותב "יראה את הצד השני", יקבל את ההערה בצורה חיובית ויתקן את הטעון תיקון.

המתח בין מפתחים לבודקים הוא במידה מסוימת טבעי ולא ניתן, לדעתי, למניעה מוחלטת. כל זמן שהוא ברמה נמוכה הוא אולי אפילו רצוי (יש מחקרים בפסיכולוגיה שמצביעים על כך שבודקים מוצאים יותר באגים אם הם מאמינים שהמוצר שנמסר לבדיקה הוא בעייתי ובאיכות נמוכה). אבל יש בהחלט מה לעשות על מנת שמתח זה לא יהפוך הרסני ומפריד אלא יעזור לשני הצדדים להשתפר.

"המתח בין מפתחים לבודקים הוא במידה מסוימת טבעי ולא ניתן למניעה מוחלטת"

"חוסר סובלנות מוביל לאיבוד ההזדמנות ללמוד אחד מהשני"

אבל אפשר גם אחרת ואפשר לעבוד על שינוי הגישה (אצלנו ואצל אחרים). הנה מקרה מהניסיון שלי שניתן ללמוד ממנו:

מפתח שלח לי מכתב ארוך שבו הוא מתלונן על רמת התייעוד

בדוח על באג. הטענה הבסיסית הייתה שהפרטים לא מספקים והצעדים לשחזור לוקים בחסר דבר שגרם לאיבוד זמן רב. עד כאן זה סטנדרטי, כל בודק מקבל מתישהו דוא"ל כזה.

מה שיידח את המכתב היה ההמשך. המפתח הסביר בפרוטרוט מה בדיוק גרם לאיבוד זמן ואיך יכולנו לשפר את התוצאה. הוא הצביע על פעולות החקירה שהוא מצפה מהבודק לעשות על מנת לבודד את הבעיה. הוא הסביר איך שינוי קל בפרטים שסופקו היו חוסכים לו זמן רב, למה הסקריפטים שהצבענו עליהם לא עובדים על המכונה שלו (ואיך לתקן את זה) ולמה היה קשה לשחזר את הבאג על פי ההוראות שבדוח. הטון לא היה של "תראה איזה בודקים גרועים אתם" אלא של "תבינו את ההשפעות של העבודה שלכם על העבודה שלי, ומתוך כך תבינו איך אתם יכולים לעשות עבודה טובה יותר".

מעבר לעובדה שהפרטים במכתב היו מועילים ונעזרנו בהם בהדרגה לשיפור רמת הדיווחים שלנו, הרי שגם ההערכה שלנו למפתח הספציפי הזה עלתה. במקום לכתוב לנו שורה קצרה וארסית של "אם היה לכם מושג איך לכתוב דו"ח נורמלי החיים של כולנו היו יותר טובים", הוא השקיע מאמץ לא מבוטל בפירוט כל מה שהיה בעייתי, איך זה השפיע עליו, ומה היה מונע או מקטין את הבעיה. אני מנחש שזה לקח לא מעט זמן לנסח את המכתב כך שגם הפרטים יהיו נכונים וגם הטון יהיה כזה שנרצה להקשיב לו. מובן שגם בהמשך הקשבנו להערות של מפתח זה בתשומת לב.

ומהצד שלנו: אם משהו שהמפתחים עושים מפריע לנו, אפשר להתעצבן, להתלונן ולחזק את הדימוי של המפתחים כקבוצה שהאיכות אינה נר לרגליה. אבל אפשר גם לנסות לשקף להם מה בדיוק הבעיה ואיך מה שהם עושים משפיע לרעה על עבודתנו. זה





טל פאר

בעל נסיון של מעל 20 שנים
כבודק ומנהל בדיקות במגוון
חברות וטכנולוגיות במודלי
פיתוח שונים. טל עובד כיועץ
עצמאי ומדריך בחברה Practical
Testing שהקים, כשהוא מלווה
ארגונים בהקמת צוותי בדיקות,
הטמעת מתודולוגיית בדיקות,
שיפור תהליכי בדיקה והדרכות
בדיקות.

טל הינו חבר בצוות המנהל של
ITCB® וגזבר ארגון ISTQB®.



אחד הדברים החשובים כשמתחילים בדיקות הוא להגדיר מהי מטרת הבדיקות (objective of testing). כמו בכל פרויקט, בעבודה או בחיים הפרטיים, אנחנו צריכים להבין מה הכוונה של צוות הבדיקות בתכנון ובביצוע הבדיקות לפרויקט המסוים ולאחר מכן להגדירו.

מתוך ניסיוני כמדריך כששואלים אנשים מהן מטרות הבדיקות שתי תשובות טיפוסיות הן "מציאת באגים" ו"להראות שהמערכת עובדת". האם לדעתך זה נכון? האם כל הבדיקות שאנחנו עושים מטרות היא מציאת באגים ולהראות שהמערכת עובדת?

השאלה של היום פשוטה.

איזה מהבאים הוא מטרה (objective) של בדיקות תוכנה?

1. כתיבת דוחות תקלה (bug reports) ברמה גבוהה
2. הערכה של איכות תהליכי הפיתוח
3. אספקת מידע לצרכי קבלת החלטות
4. וידוא שיש נֶעֱקְבוּת (traceability) בין הדרישות לבדיקות



*לצפייה בתשובה המפורטת - דפדפו לעמוד 10

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 לאיריס קוטליצקי:

info.testingworld@gmail.com



על החברה

SentinelOne נוסדה בשנת 2013 ע"י קבוצת ישראלים מומחים באבטחת מידע. במשרדים בת"א נמצאת קבוצת המחקר והפיתוח שמפתחת את הדור הבא של הגנה על מחשבי קצה ושרתים. מוצר התוכנה משתמש במספר שכבות הגנה הכוללות ניתוח התנהגותי (למידת מכונה על המידע שנאסף), מלכודות ועוד, בכדי לעצור התקפות Zero-day (מתקפה המנצלת חולשה שעדיין לא התגלתה) שמוצרים אחרים לא מסוגלים. בנוסף המוצר מספק מידע שקשה להתחרות בו על ההתקפות, כשהוא משפיע בצורה מינימלית על המערכת.

על המוצר

SentinelOne מפתחת את הדור הבא של האנטי-וירוס. המוצר המותקן על מחשבי הקצה כולל מספר שכבות הגנה שמנטרים ללא הרף אירועים וסורקים קבצים. כאשר המערכת מזהה פעילות חשודה היא מתריעה ונכנסת לפעולה לפי המדיניות שהלקוח הגדיר. כיום החברה מספקת הגנה למחשבי קצה בסביבת Windows, Mac ולינוקס.

במאמר זה נתמקד במעבדת הביצועים הפועלת בסביבת החלונות והמוצר אותו נבדוק מורכב ממספר תהליכים הרצים על המערכת.

הרקע להקמת המעבדה

כשהחברה הייתה בתחילת דרכה והמשאבים הכספיים וכוו האדם היו מצומצמים, באופן טבעי המיקוד הופנה בעיקרו לפיתוח יכולות ופחות בהשקעה בדברים "שמסביב". אך מאוד מהר נוספו לקוחות גדולים ונדרשה אחריות גדולה יותר. מכיוון שהתוכנה שלנו מותקנת על המחשב ומנטרת אחר כל פעילות המשתמש, עלה הצורך לבדוק את התנהגות המוצר ולכמת את ההשפעה שלנו על המערכת לעומת מערכת ללא המוצר.

התחלנו בחיפוש אחר תוכנות מדף הקיימות בשוק. לא מצאנו פתרון מתאים עבור דרישותינו ולכן החלטנו להקים מעבדה.

המטרה

המעבדה הוקמה מקו האפס וכמו בכל משימה היה חשוב להגדיר את המטרה:

1. כמה המוצר משפיע על מחשב הקצה לעומת מערכת "נקיה"
2. השוואת ביצועים בין גרסה לגרסה
3. מדידה בין builds שונים במטרה לבדוק האם הכנסה של קוד ספציפי שיפרה או הרעה את הביצועים

הדרך

התחלנו עם הגדרות המדדים שאנו רוצים לאסוף, מה אנחנו רוצים לנטר ואיך לבצע מדידות.

המשכנו בבניית "תסריטים" לבדיקות, כל תסריט כזה הוא בעצם סט פעולות על המחשב המדמה פעילות של משתמש, שרת ועוד.

לדוגמה, תסריט "משתמש רגיל" הכולל פעילות קבצים (העתקות/העברות ועוד), התקנות של תוכנות "כבדות" (כמו Office) וקומפילציה של קוד.

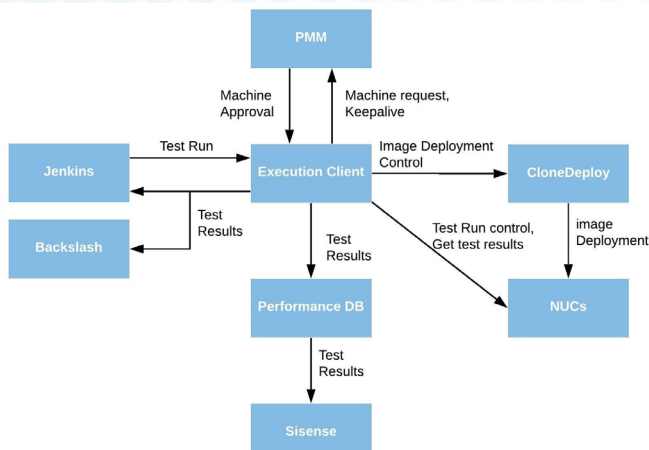
בהמשך הוספנו עוד תסריטים - מדידת זמני עלייה של המערכת, זמן פתיחת תוכנות שונות.

נסביר את המעבדה בשלושה חלקים: תשתית, איסוף נתונים וניתוח נתונים.



תומר אלון

בן 38, נשוי +2. מהנדס בדיקות בחברת SentinelOne שנים. בהכשרתי אני מהנדס חשמל ואלקטרוניקה אבל נמשכתי יותר לתחום התוכנה ואיכשהו התגלגלתי לתחום בדיקות התוכנה והתאהבתי במקצוע.



תשתית

Jenkins: מתחיל את התסריט ושומר מידע כללי על הריצה

EC (Execution client): שרת שמריץ את הבדיקה ואחראי על פעולות השליטה

Mini-PC: NUC שהתסריט רץ עליו, ה-DUT שלנו

PMM: מחזיק ומעדכן את מצב המכונות לניהול של משאבים

CloneDeploy: אחראי על פריסת האימג'ים למכונות

מהרגע הראשון הבנו שאת התסריטים צריך להריץ על מכונה אמיתית ולא על סימולציה (Virtual). למרות הפשטות של הפריסה אין דרך קלה להבטיח משאבים שתלויים בשרת שמריץ את ה-VM. לכן רכשנו כמה מכונות Intel Nuc שהן קטנות (משיקולי מקום) וגם יחסית זולות. כמובן שהחומרה חייבת להיות זזה בכל אחת מהמכונות בשביל לא להשפיע על המדידה. חילקנו את ה-NUCs לשתי קבוצות: כאלו המכילות דיסק קשיח מסוג SSD (כונן קשיח מהיר) וכאלו המכילות HDD.

המשכנו בבניית Images שיפרסו בתחילת כל תסריט על ה-Nuc. היינו צריכים לדאוג לנטרל כל מיני רעשי רקע שעלולים לצוץ כמו עדכוני מוצר (של חלונות ואחרים), תהליכים שגרתיים של המערכת שעולים בצורה מתוזמנת וכו'. כמובן שבריצות הראשונות גילינו פערים בתוצאות בין ריצה לריצה. ניתחנו כל פער מה וניטרלנו תהליכים. לאט לאט הצלחנו להגיע לייצוב של המערכת כך שכל ריצה עם אותם פרמטרים הראתה תוצאה זזה (או כמעט זזה) לקודמתה.

בשביל לנהל את ה-Images ולפרוס אותם השתמשנו במערכת קוד פתוח CloneDeploy. המערכת מסוגלת לפרוס Image נקי לכל מכונה בזמן קצר יחסית למספר מכונות במקביל.

כל Image מיועד לתסריט אחד או מספר תסריטים ומכיל את כל הקבצים הדרושים לתסריט על מנת להקטין את זמן ההכנה לריצה.

כל ההרצות מנוהלות ע"י מערכת Jenkins CI שמכילה מספר Jobs ישנם Jobs שניתן להריץ באופן ידני ע"י בחירת מספר פרמטרים כמו Branch, Build, תסריט ועוד. אבל את עיקר ההרצות מבצעים ע"י Jobs אוטומטיים שמתוכננים לרוץ בשעות הלילה (השעות האלו נבחרו מכיוון שבזמן זה הרשת פחות עמוסה ובכדי שהן יסתיימו עד שעות הבוקר שאז אפשר לנתח את הנתונים). כל תסריט ירוץ 4 פעמים (בשביל לקבל ממוצע) על כל סוג של דיסק HDD/SSD.

בכדי שלא נתחיל ריצה על מכונה לא פנויה נדרשנו לניהול משאבים ולכן יצרנו מאגר של מכונות (Physical Machine Manager - PMM).



הרצת בדיקה:

1. מתחיל בהמתנה למכונה פנויה במאגר, אם מצא מסמן אותה "כלא פנויה"
2. לאחר שסימן מכונה, פונה לשרת ה-Clone שפורס על המכונה את ה-Image הנכון לפי סוג התסריט
3. בסיום הפריסה המכונה עולה וה-EC מעתיק את תסריטי הבדיקה
4. מריץ את התסריט ובודק כל חמש דקות אם הריצה הסתיימה
5. כשהבדיקה מסתיימת:
 - a. ה-EC אוסף את התוצאות ומעבדן ל-Jenkins ואת קבצי המונויטור מעתיק ל-DB
 - b. משחרר את המכונה במאגר

הקמנו מעבדת ביצועים לפי דרישות החברה ובהתאמה למוצר בעזרת כלים שונים, חלקם פיתחנו במקום וחלקם רכשנו כלים של צד שלישי. ביצענו אינטגרציה לכל הרכיבים האלו והמעבדה שלנו רצה באופן אוטומטי.

איסוף נתונים

במקביל לריצה של התסריט, ישנו גם סקריפט שרץ ואוסף במקביל נתונים שיעזרו לנו לקבוע את נתוני הביצועים. אנו אוספים את הנתונים הבאים כל 10 שניות:

- זיכרון, CPU של המערכת
- קריאה וכתיבה לדיסק - IO
- ניצול של הרשת קבלה ושליחה - Bytes
- זיכרון CPU-עבור כל Service שהתוכנה שלנו מריצה

בנוסף בסיום התסריט אנחנו מנתחים את הלוגים שיוצרת התוכנה הנבדקת ושואבים משם נתונים נוספים.

את כל המידע הזה אנחנו מעבירים ל-DB בסיום התסריט.

המידע שנאסף מכיל 50 מדדים שנאספים פעם ב-10 שניות.

בתסריט שאורך שעה אנחנו נאסף 18,000 מדדים.

אנחנו מריצים כ-100 הרצות בלילה שנותנות כ-2,000,000 שורות ב-DB.

מכיוון שהתוכנה שלנו מותקנת על המחשב ומנטרת אחר כל פעילות המשתמש עלה הצורך לבדוק את התנהגות המוצר ולכמת את ההשפעה שלנו על המערכת לעומת מערכת ללא המוצר.

ניתוח נתונים

כאמור יש לנו כמויות עצומות של מידע לנתח מכל ריצת לילה ובנוסף יש את המידע שקיים לגבי ריצות קודמות, כך שבכל רגע נתון יש עשרות מיליוני מדדים ב-DB. ניתוח של כמות כזאת של מידע נעשה בעזרת תוכנת BI (Business Intelligence) שיכולה לבצע פעולות מתמטיות וסטטיסטיות באופן יעיל ולהציג את המידע המעובד בצורה גרפית בתצוגות שאנו קובעים. למשל, תצוגה מתאימה למנהלים שמציגה רק את המידע המקוצר ותצוגה אחרת שנותנת מידע מפורט יותר לאיש הבדיקות.

אנחנו בחרנו להשתמש בתוכנת צד שלישי של חברת Sisense. התוכנה מאפשרת לבצע ETL (Extract, Transform and Load). השרת מתחבר ל-DB ושואב משם את הנתונים, לאחר מכן הוא

מעביר אותם למודל שבנינו ומבצע פעולות חישוביות שונות, כמו ממוצעים. את כל החישובים הוא מאחסן מקומית. כאן מגיע השלב שאנחנו בונים תצוגות שונות או Dashboards על בסיס נתונים אלו.

למשל, אנחנו יכולים לפתוח כל בוקר את ה-Dashboard המתאים ולראות את המצב של הגרסה האחרונה לעומת הגרסאות האחרות ואם יש בעיה בביצועים המחייבת בדיקה נוכל לבדוק בקלות ויותר לעומק ב-Dashboard.

אחד הדברים המאתגרים זה לקבוע מה נחשב לפגיעה בביצועים ואנחנו צריכים להתמודד עם שאלות כמו:

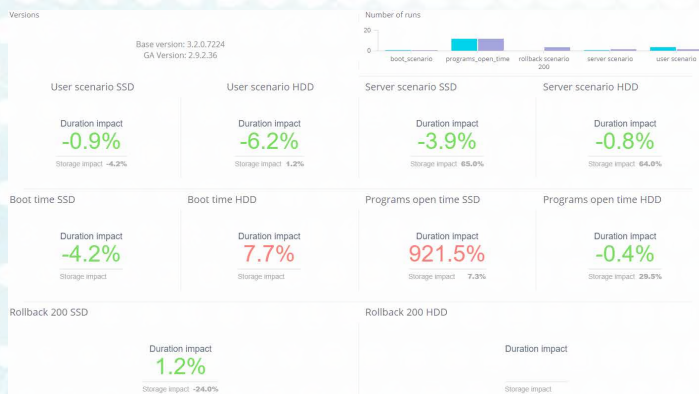
- האם עלייה ב-CPU אבל ירידה בזמן הביצוע נחשב פגיעה או שיפור?
- כמה עלייה בזמן הביצוע נחשבת כפגיעה?

ולכן קבענו KPI (Key Performance Indicator) המהווים את הסף ממנו אנו מזהים אם יש בעיה. לדוגמה: אם שלב כלשהו בתסריט ישנה חריגה של מעל ל-7% בזמן ריצה לעומת גרסה אחרת אז אנחנו צובעים את התוצאה באדום ושולחים בנוסף התרעה לידיע שיש כאן בעיה. כך שהמערכת פועלת באופן כמעט אוטומטי: ריצה כל לילה על גרסה אחרונה, איסוף התוצאות, עיבוד ודיווח על חריגות.

הקמנו גם Dashboard לשימושם של המפתחים שיוכלו להשוות בין גרסאות שונות. המטרה היא שמפתח יוכל לבנות Build עם השינויים שלו, להריץ מעבדה על הגרסה הזאת ואז לנתח את התוצאות מול גרסה בלי השינויים כך שבקלות יוכל לזהות הרעה או שיפור בתוצאות.

בנוסף אנו מנתחים במערכת תוצאות חריגות. כמו שכולנו מכירים, דברים לא תמיד עובדים כמו שצריך וריצות עלולות להיכשל מכל מיני סיבות ולכן אנחנו מנתחים גם תוצאות חריגות. לדוגמה, אם רצו 4 בדיקות על אותו תסריט ובאחת מהתוצאות קיימת תוצאה שהיא חריגה מהממוצע של כל שלושת הריצות אנחנו נוכל לבדוק מה קרה באותה ריצה ובמידת הצורך לנטרל את ההשפעה של הריצה הזאת על המדידות.

במסך הזה ניתן לראות במבט כללי השוואה בין הגרסה האחרונה שנמדדה בלילה לעומת גרסה שנמצאת אצל הלקוחות:



רואים כאן כל תסריט שרץ ואת ההבדלים באחוזים בין הגרסאות.

*כל התמונות מציגות תוצאות משלב הפיתוח ולא משקפות את מצב המוצר



סיכום

הקמנו מעבדת ביצועים לפי דרישות החברה ובהתאמה למוצר בעזרת כלים שונים, חלקם פיתחנו במקום וחלקם כלים של צד שלישי. ביצענו אינטגרציה לכל הרכיבים האלו והמעבדה שלנו רצה באופן אוטומטי.

כל לילה Job אוטומטי לוקח את גרסת התוכנה האחרונה ומתחיל להריץ עליה את המעבדה. בסיום כל ריצה התוצאות מועברות ל-DB. לאחר מכן מופעלת מערכת ה-BI ע"י Sisense שלוקח את הנתונים, מעבד אותם ושולח התרעה אם ישנה חריגה. כך שהמערכת היא אוטומטית לחלוטין למעט כמובן תקלות שקורות פה ושם.

בנוסף, ניתן לקבל מידע מעמיק יותר מהמעבדה וכך לזהות חריגות ולדלות יותר נתונים.

המעבדה משמשת כרגע בעיקר את צוות הבדיקות, המידע שמופק משמש את מנהלי הפרויקט להחלטות אם הגרסה מוכנה לשחרור או לא. שימוש נוסף שנעשה זה השוואה בין בילדים שונים כאשר מפתח מוסיף או משנה משהו במוצר ומעוניין לבדוק את ההשפעה של השינוי מבחינת הביצועים. הוא פשוט מריץ את המעבדה על הבילד לפני השינוי ועם השינוי והתוצאות יוצגו לו בצורה נוחה.

כך השגנו את המטרות שהוצבו בתחילת הדרך. אנחנו יכולים לכמת את ההשפעה של המוצר שלנו על המערכת לעומת מערכת "נקיה", להשוות בין גרסאות שונות של המוצר ולזהות את ההשפעה של שינויים בין בילדים שונים על ביצועי המערכת.

בעתיד אנחנו מתכננים להוסיף מעבדת ביצועים שתרוץ על מוצרים נוספים כמו תוכנת ההגנה שמותקנת על MacOS.

במסך הזה ניתן לראות יותר לעומק תוצאות לעומת כל הגרסאות הקודמות של המוצר אם יש חריגה מה-KPI היא נצבעת אדום:

Benchmark

3.2.0.7224

Scenario	Disk Type	Runs	2.5.4.104	2.6.4.5961	2.6.5.5979	2.7.4.6510	2.8.2.36	3.1.1.12
boot_scenario	hdd	1		2.57%	91.26	3.44%	90.55	97.71
	ssd	1		996.77		2.58%	925.36	946.18
programs_open_time	hdd	12	807.38%	815.03	880.7%	807.38	821.25%	775.78
	ssd	12	4.236.19			1.162.21	9.29%	1.171.2
rollback_scenario_200	hdd	4	4.944.9	4.957.69		4.886.06		4.932.96
server_scenario	hdd	1	4.146.59	4.170.19		4.158.36		4.124.33
	ssd	2		14.16%	5.969.30	15.12%	5.932.90	7.277.67
user_scenario	hdd	4	5.159%	4.914.02	4.972.23		4.981.09	4.907.28

Scenario	Disk Type	Runs	2.5.4.104	2.6.4.5961	2.6.5.5979	2.7.4.6510	2.8.2.36	3.1.1.12	3.1.2.17
programs_open_time	hdd	12	25.23%	16.1	27.65%	19.2	20.52%	16.9	22.4
	ssd	12	26.17%	17.3	27.24%	16.5	2.24%	23.1	25.6
rollback_scenario_200	hdd	4	1.051.4	1.052.6	1.067.9	1.067.9	1.063.3		921.1
server_scenario	hdd	1	1.051.4	1.143.6	77.69%	36.1	63.96%	36.1	
	ssd	2	131.22%	27.0	76.38%	34.9	65.01%	37.9	63.77%
user_scenario	hdd	4	32.85%	725.1	19.59%	945.1	1.22%	994.6	975.8

השגנו את המטרות שהוצבו בתחילת הדרך: אנו יכולים לכמת את ההשפעה של המוצר שלנו על המערכת לעומת מערכת "נקיה", להשוות בין גרסאות שונות של המוצר ולזהות את ההשפעה של שינויים בין בילדים שונים על ביצועי המערכת.



בחן את עצמך | טל פאר

הפתרון לשאלה

השאלה שלנו מתייחסת לפרק 1.1 בסילבוס שמדבר על יסודות בדיקות התוכנה.

בפרק מוגדרות מספר מטרות אופייניות של בדיקות שאחת מהן מוזכרת בשאלה והיא התשובה הנכונה. נגיע לזה בהמשך. אחדות מהמטרות האופייניות לבדיקות הן ידועות ונחשבות "טריויאליות", למשל מציאת כשלים (failures) ופגמים (defects). אחרים יגידו שמטרות הבדיקה היא הן אימות (verification) הדרישות, שזה נכון. אנחנו רוצים לאמת שיכולות המערכת עונות על דרישותיה. אך מה לגבי תיקוף (validation) הדרישות? כלומר, מה לגבי בדיקה שהמערכת עונה על צרכי המשתמש? הרי מערכת יכולה לתפקד בצורה טובה אך עדיין לא לענות על צרכי המשתמש.

בהמשך הסילבוס מוצגות רמות הבדיקה (test levels) השונות ולכל אחת מהן מטרות משלה. אחת ממטרות בדיקות הרכיבים (component testing) היא וידוא ההתנהגות הפונקציונלית של הרכיב בהתאם לאפיון, בעוד שמטרת בדיקות האינטגרציה (integration testing) היא לוודא שהממשק בין הרכיבים מתפקד בהתאם לאפיון.

הבה נבחן את התשובות

תשובה נכונה. אחת המטרות המרכזיות של בדיקות היא לספק מידע לבעלי העניין (למשל, מנהל הפרויקט) כדי שיוכל לקבל החלטות. בלי מידע על תוצאות הבדיקות, יתקשו בעלי העניין לדעת מה מצב המערכת, אם יש בה תקלות שלא נפתרו, אם הסיכונים שזוהו נבדקו ורמתם ירדה. בעזרת המידע שהבדיקות מספקות ניתן להחליט אם לשחרר את המערכת ללקוחות או לא.



תשובה לא נכונה. נכון שיש להקפיד שדוחות התקלה יכתבו ברמה גבוהה ויספקו את מירב המידע על התקלה. אך דוחות אלה הם תוצר של הבדיקות ולא מטרה שלהן.



תשובה לא נכונה. אחת ממטרות הבדיקה היא להעריך את איכות המוצר אך לא את איכות תהליך הפיתוח. אולם, כתוצאה מתוצאות הבדיקה אפשר להשפיע על תהליך הפיתוח. למשל, אם יתגלה שמודול אחד במערכת מכיל הרבה יותר תקלות ממה שציפינו (למשל על ידי ניתוח סיכונים), נוכל לנתח את תהליך הפיתוח ולשפר אותו כדי למנוע תקלות בעתיד.



תשובה לא נכונה. נעקבות היא כלי שיכול לספק מידע על האפקטיביות של הבדיקות, כשצוות הבדיקות יוכל להראות כיסוי של דרישות על ידי בדיקות. כך נוכל להראות כמה דרישות כוסו על ידי בדיקות שעברו, כמה על ידי בדיקות שנכשלו וכמה מכילות באגים.



עוז צופי

"מוצר שעבר בדיקות,
זה מוצר שהשתמש לא
שם לב שאין בו בעיות"

בן 38, מהנדס בדיקות, בעל
6 שנות ניסיון.

מנהל מחלקת בדיקות
ב-Action item solutions,
חברת פרויקטים בת"א.

אוהב לחקור את המובן
מאחורי, לשאול שאלות
שמובילות לעוד שאלות,
ולגלות עולמות חדשים ליד
הבית.

אחד התחומים המעניינים ביותר בעולם פיתוח התוכנה בשנים האחרונות הינו פיתוח Web. כחלק מעולם של פיתוח בקוד פתוח, הוא נגיש לכולם ומותאם לכל עולם Web והינו פתרון אידיאלי לקהלים רחבים. גם טכנולוגיות חדשות ושיפורים מקלים את הפיתוח בו, ואפשר להגיד ללא עוררין שהוא הבסיס החדש למגוון רחב של אפשרויות פיתוח, ועל כן, מצריך גם תהליך בדיקות מסודר.

בדיקות Web, הינן מגוונות וניתנות לשינויים בין משתמש למשתמש יותר מאשר תחומי בדיקות אחרים. כמו בדיקות מובייל, המוצר הסופי שלנו יכול לרוץ על מגוון רחב של מכשירים (בעידן בו מחשבים אישיים וטלפונים אינם הפלטפורמות היחידות לשימוש באינטרנט), גדלי מסכים, אפשרויות קלט, ועוד. את ההמלצות שלי במאמר זה אני מדגים על **דפדפן כרום**, שכן הוא דפדפן מאוד מודרני שמתעדכן בתכיפות ומוביל באחוז השימוש שלו ביחס לדפדפנים אחרים.

1 להשתמש באנליטיקות

מכיוון שישנם מספר גדול של מכשירים שאפשר לבדוק את תקינות המוצר שלנו, אנו נרצה קודם כל להבין איך הלקוח שלנו משתמש במוצר, ורק לאחר מכן לכתוב לו את הבדיקות. לדוגמה: אם המוצר שלנו הינו אתר אינטרנט, נרצה להבין האם המשתמשים יגלשו בדפדפן בדסקטופ או גם בנייד, ואם כן, באילו מכשירים, באיזה מערכות הפעלה, ובאיזה רזולוציות מסך. הדבר יכול להיות מיותר אם אנחנו בודקים עמוד שיהיה מוטמע בתוך מערכת חומרה סגורה שלא משנה את האיפיונים שלה.

2 תמיד לבדוק עם הגרסה האחרונה של הדפדפן

בניגוד למערכות הפעלה שהמשתמשים לא בהכרח ממהרים לעדכן (למעט iOS), המשתמשים נוטים לעדכן את הדפדפן שלהם לגרסה האחרונה. לכן רצוי לבדוק תמיד עם הגרסה האחרונה או אפילו בגרסה בטא שעתידית לצאת.

3 להסתכל תמיד מאחורי הקלעים

מומלץ תמיד לבדוק עם כלי מפתחים פתוח (F12), ולראות מה קורה מאחורי הקלעים בזמן שהאתר שלנו רץ ברקע, כך נוכל ביתר קלות לתעד את הגורם לבעיה, ולתת למפתח עוד מידע רלוונטי איך לשחזר אותה.

4 שגיאות הם באגים ולא פיצ'רים - תמיד

ניתן להכנס ל-F12 ותחת הקונסול לראות מסרים אשר מתקבלים מהדפדפן, שגיאות/באגים יצבעו באדום. אחריות על באגים נעשית בידי הפיתוח ולכן חשוב לזכור להציג לאיש הפיתוח מידע זה בדיווח הבאג על מנת לאשש זאת. לרוב שגיאות אלו נראות חסרות חשיבות, שכן המוצר שלנו יכול לעבוד בצורה חלקה עם חלקן, אך עדיין חשוב להציף את המידע ולרוב לפתוח באג. לידע כללי, אזהרות הינן שגיאות המוצגות בצהוב ולרוב אינן באגים, אך רצוי להתייעץ לגביהם.



5 להשתמש בכלי ביקורת

כלי חינוכי שניתן יחד עם כמעט כל דפדפן, ניתן להגיע אליו על ידי F12 > Audits. הרצה של הביקורת על האתר שלנו, עשויה להניב מספר באגים מהירים ואיכותיים.

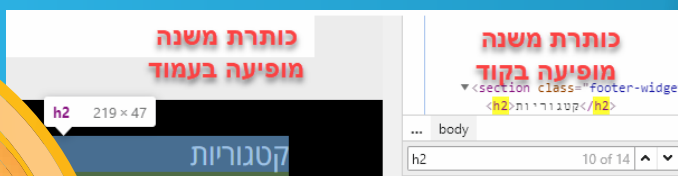
6 תמיד לזכור לשמור ולא לשכוח

תחת F12 > Networks, אנו יכולים לראות את התקשורת בין האתר לדפדפן שלנו, האפשרות Preserve log, הינה במצב ברירת המחדל אינה דלוקה. מומלץ להדליק אותה על מנת שהלוגים לא ימחקו לנו ונוכל להתמקד בצעדי השחזור של הבאג ולשלוף באגים בהמשך.



7 חיפוש בתוך הקוד

לפעמים יכול להיות הרבה יותר קל לחפש משהו בעמוד אינטרנט, אך לרוב אלמנטים תכנותיים רבים אינם מוצגים למשתמש, ומציאתם יכולה לקחת לנו זמן רב. על כך ניתן לבצע חיפוש של אלמנטים בדפדפן. תחת F12 > Elements, נקיש Ctrl+F, יפתח לנו שדה חיפוש שדומה לשדה החיפוש בעמוד האינטרנט אך כאן נוכל לחפש שמות של אלמנטים, ID, פרמטרים בעמודי HTML וגם CSS הקשורים לעמוד שאנו עומדים עליו. לדוגמה: נניח שיש לנו אלמנט בקוד של כותרת משנה, לדוגמה 'h2', חיפוש של ערך זה באתר עצמו לא יניב לנו תוצאות רלוונטיות, אבל חיפוש באלמנטים יציג לנו את כל הכותרות משנה הללו בעמוד הנוכחי.





ניצן גולדנברג

מזה 5 שנים בתחום בדיקות התוכנה. מהנדס בדיקות בחברת [SeatGeek](#), המוביל הראשי של קבוצת המיטאפ [TestIL](#), עורך וכתב במגזין "עולם הבדיקות", מרצה בקורסים לבודקי תוכנה וחבר בצוות המייעץ AB של ITCB®.

- 13 מפתחים
- בודק אחד
- 4 מנהלי מוצר
- איש תשתיות
- מנהל ה-R&D
- 5 אנשי שירות

ביום ציד הבאגים, אני לא הייתי בודק פעיל, אלא מנחה של ציד הבאגים. ראוי לציין שבגישה של "ציד הבאגים" יש גם יתרונות וגם חסרונות.

אלו היתרונות העיקריים:

1 גיוון - איכות, כמו שהיפוי
הוא בעיני המתבונן? גם לנו יש נטיות והעדפות וזאת על ידי איסוף דעות של רבים, יכולנו להתרכז בבעיות שנמצאו שקרוב לודאי היו משפיעות על הלוקוחות שלנו.

2 כיסוי - בשל הכמות
הגדולה של משתתפי הציד הספקנו לבצע עבודת בדיקות שלוקחת שבועיים וחצי בזמן של 4-5 שעות

($4 \times 25 = 100$). ברור שזה מסוכן, (לפי המאמר של ג'יימס באך ["Two Scoops of Bugs"](#)) הרי לא כל הבאגים הם באמת באגים וכל משתתפי הציד הם לא בודקים באמת אבל הצלחנו להטיל רשת רחבה יותר על שטח גדול יותר. אין לי אפשרות לדעת איפה יהיו חורים ברשת שלנו אבל זה נתן לנו טיפים של היכן יכולים להימצא באגים ולחקור זאת.

שקט נפשי למנהלי המוצר - מנהלי המוצר היו מעורבים בציד הבאגים ויכלו לראות מקרוב את היכולות של המוצר שלהם.

שקט נפשי עבור המפתחים - מפתחים מכירים את תחומי המוצר שהם חוששים ממנו הכי הרבה (מבחינת הסיכון לאיכות) וזה נתן להם את ההזדמנות לחקור את הסיכונים האלה פוטנציאלית.

בדיקות משתמשים - משתמשי הקצה שלנו (או לקוחות משלמים) כבר היו מיוצגים על ידי מנהלי המוצר, אבל המשתמשים הפנימיים שלנו (צוות התמיכה ויועצים) היו מיוצגים הרבה יותר במהלך ציד הבאגים ודעתם הייתה חשובה להערכת איכות המוצר.

כמו בכל הדברים הטובים היו גם חסרונות:

1. כל באג היה צריך להיבדק במהירות כדי להתאים אותו לדיווח באגים רגיל, כך שניתן היה להבין אותו לאחר ציד הבאגים דבר אשר לקח זמן רב ונתפס כיתר פדנטיות.
2. כל באג היה צריך להיבדק ולשחזר כדי לראות אם זה היה בעצם באג (כלומר האם זה מאיים על הערך של המוצר לבעלי עניין), שוב זמן רב מאוד.
3. היעדר באגים המדווחים על שטח המוצר אינו אומר שאין באגים.
4. השתמשנו באנשים שלא היו באמת בודקים ולכן איכות הבדיקות היו בסופו של דבר בסימן שאלה.

יום הציד

רוב הצוותים נמצאים באותו משרד אבל חלק מהצוותים היו אנשים שעבדו מהבית בדרך כלל ועשו את המסע למשרד במיוחד באותו היום. הוחלט להתחיל את היום בשעה 10:00 כדי לתת לכולם להגיע בינוניות למשרד. הוחלט כי באותו היום כולם יכולים להגיע בלבוש חופשי ולא אלגנטי ואפילו מנהל ה-R&D הסכים להזמין את כולם לארוחת צהריים מרוכזת.

תומך טכני בבית תוכנה הייתה המשרה הראשונה שלי בעולם ההיי-טק, המנכ"ל בראיון אהב את הרעיון שבדיוק סיימתי קורס בדיקות ובגלל שלא הייתה לו מחלקת QA הוא החליט לתת לי להרים את כל מערך הבדיקות מההתחלה ובכך נתן לי את ההזדמנות להיכנס לעולם הבדיקות תוכנה. זו הייתה המשרה הראשונה שלי כבודק תוכנה ובנוסף הייתי צריך לתכנן את כל מערך הבדיקות ולא ידעתי מה לעשות אז התחלתי לחקור ברשת על תהליכי בדיקות.

בשלב זה אני קורא איפשהו (למרבה הצער אני לא זוכר איפה) על הרעיון של Bug Hunt - "ציד באגים" וזה תפס את הדמיון שלי והחלטתי לשלב את זה ברשימת תהליכי הבדיקות.

המנהל שלי אהב את הרעיון, הסכמנו שכאשר נתקרב לשלב שחרור הגרסה נריץ "ציד באגים" אחד לניסוי.

כחודש לפני שחרור הגרסה התחלנו לתכנן ולהתארגן לציד הבאגים הראשון שלנו.

בכדי שיהיה מספיק זמן למפתחים לטפל בבאגים שימצאו בציד הבאגים ובכדי שהגרסה (build) של המערכת תהיה יציבה מספיק, הוחלט שהיום המיועד יהיה 3 שבועות לפני התאריך הרישמי של שחרור הגרסה. היום המיועד נרשם ביומנים של כל עובדי החברה, הוקצו חדרים לכל הצוותים, נפתחו משימות במערכת ניהול זמנים, ההתרגשות החלה וכך גם החשש שהרי אנו עומדים לשתף את הבדיקות עם עמיתים שלנו שאינם בודקים ביום יום. כמו כן, הכנתי את כל המידע שאספתי במהלך התקופה שלי בחברה כדי לסייע בשיתוף הבדיקות הזמני בניסיון למנוע אנרכיה בשורות ולכן עשינו את הדברים הבאים:

- פרסמנו כמה מדריכי "כיצד" באתר הוויקי של החברה המסבירים כיצד לכתוב דוחות באגים ברמת בודק וטכניקות בדיקה אפשריות
- יצרנו פרויקט מיוחד במערכת דיווח הבאגים שלנו, פישטנו את טופס דיווח הבאגים שיהיה קל להבנה על ידי אלו שלא פותחים באגים בכל יום
- חילקנו את המערכת לאזורים בעזרת מנהלי המוצר ויצרנו תוויות לכל אזור כדי לסמן באיזה אזור נמצא הבאג
- יצרנו מאגר באגים קיימים כדי שלא יהיה מצב של כפילויות
- הקמנו סביבת בדיקות ייחודית שכללה התקנת המערכת ואינטגרציות הרלוונטיות לה, העלנו בסיס נתונים עם מידע רב ככל האפשר, יצרנו משתמשים לכל איש צוות
- הכנו לו"ז מסודר ליום הבדיקות

היו לא מעט באגים שנדחו אבל כמו ציידים אמיתיים הם החלו ללמוד מן הטעויות שלהם

אז... מה זה "ציד באגים"?

לפי דעתי והבנתי Bug Hunt זה כמו שנשמע, ציד באגים. חשוב לי לציין שזו לא הייתה הגישה היחידה לבדיקות באותה גרסה או הדרך היחידה למציאת באגים. הפעילות הייתה חלק ממשימת הבדיקה שלי אשר הייתה לספק מידע למנהלי המוצר כדי לסייע להם לקבל החלטה לגבי שחרור הגרסה, היינו מעוניינים למצוא באגים רבים ככל האפשר בתוך זמן קצר כך שנוכל לתת עדיפות לבאגים הקריטיים לפני שחרור הגרסה.

מי מעורב ב"ציד באגים"?

צוות בדיקות ייעודי היה רעיון חדש בחברה שלי. הייתי הבודק הראשון ועברתי לא מעט "כאבים" כדי לערב את העמיתים משאר המחלקות בחברה. בזמן ציד הבאגים התחלנו עם 6 אנשים בצוות (אני הבודק, 2 מפתחים, 2 אנשי שירות ומכירת מנכ"ל). ברגע שהתחלנו את יום הבדיקות התחילו לראות עוד אנשים בחברה שאנו נהנים כך שלבסוף סיימנו עם 25 אנשים





התוצאות

היום של אחרי ציד הבאגים היה מעיקרו מיועד לבחינת הבאגים בפירוט רב יותר, זה היה כרוך בשחזור הבאגים, הבהרת כל הצעדים, קיטלוג הבאגים כל שניתן היה לתעדף כל באג ובאג מול מנהלי המוצר.

בסוף יום ציד הבאגים רשמנו כ-180 באגים, לאחר בחינת כל הבאגים נשארו לנו 90.

רבים מהבאגים שהועלו במהלך ציד הבאגים סווגו כבעלי עדיפות נמוכה אבל בנוסף היו כ-10 באגים שנחשבו לקריטיים כך שצוות הפיתוח שלנו עבד בשארית הספרינט על תיקונם.

מסקנות ולקחים

ציד הבאגים זכה להצלחה ענקית בארגון ממספר סיבות:

- התאפשר לנו להסתכל על המוצר שלנו בפרספקטיבות שונות
- כל צייד באגים הרגיש שייכות והיה מעורב בתהליך ובסופו של דבר לקח קצת בעלות על איכות המוצר
- צייד הבאגים חוו כמה מהאתגרים העומדים בפני הבודק על בסיס יומיומי
- למפתחים שעבדו על המוצר הייתה הזדמנות לחקור את תחומי המוצר שהם היו מודאגים לגביהם
- מנהל המוצר יכול היה "להרגיש" את המוצר ואת הבעיות הקיימות בפעם הראשונה (לא במצב של דמו)
- הצלחנו לבדוק עומס אקראי נגד המוצר שלנו על ידי מתן אפשרות לכל צוות לבצע בדיקות עצמאיות שלהם זה מזה
- זיהינו פערים בין מה שמנהל המוצר ביקש לבין מה שפותח בפועל
- זה נתן לבעלי העניין שלנו תחושה חמה שההשקעה שלהם השתלמה, עם מפת דרכים ברורה עבור העבודה שנותרה לפני שחרור גרסה

בעוד אלה הם כל היתרונות הגדולים, היו כמה לקחים חשובים שנלמדו:

- גישה זו נוטה לטובת כמות על פני איכות במונחים של באגים, כלומר נמצאו כמויות גדולות של באגים שזה טוב אבל אחוז מאוד גבוה מהדיווחים היו מאוד בעייתיים לשחזור בשל הרמה הנמוכה של הדיווח שכן רוב משתתפי הציד היו אנשים שלא פותחים באגים ביום יום.
- בדיקת באגים כנגד תבנית דוח באגים סטנדרטית היא משימה חסרת תוחלת ואינה מבטיחה כי הבאג הוא: תקף / ניתן לשחזור / סימפטום או שורש הבעיה.

בסופו של דבר ציד הבאגים הלך ממש טוב ואנחנו בהחלט יותר מלוכדים כצוות מאשר קודם וניתן היה לראות שהמשימה העיקרית שלנו היא למצוא באגים חשובים תוך זמן קצר או כדי להדגים את יכולת המוצר לבעלי העניין המרכזיים בחברה.



התחלנו את היום בתדרוך אשר העברתי על ידי מצגת המסבירה איך כל היום הזה הולך לקרות אשר הנושאים היו:

- הפורמט והציפיות של היום
- כיצד להתחבר לסביבת הבדיקות
- אזורי המוצר שמנהלי המוצר החליטו שיכוסו
- כיצד לדיווח באגים
- טכניקות בדיקה מוצעות
- והדבר הכי חשוב, הפרסים שאנו מחלקים לפי קריטריונים שנקבעו מראש

זיהינו פערים בין מה שמנהל המוצר ביקש לבין מה שפותח בפועל

חילקנו את היום לשני סשנים שנמשכו כשעתיים וחצי כל אחד כדי לנסות לשמור על ריכוז אופטימלי וגם שיהיה זמן להתארגנות ומנוחה.

משתתפי ציד הבאגים חולקו לצוותים של שניים, חלק אחד יהיו ה"ציידים" ויחפשו את הבאגים והחלק השני יהיו המובילים אשר יאספו את המידע ויפתחו את הבאגים, הכוונה הייתה שהם יחליפו תפקידים אחרי ההפסקה.

במשך כל המפגשים היינו מקרינים על מסך ענק את ספירת הבאגים (כדי לחזק את הרוח התחרותית) ואת מפת האזורים אשר נמצאו בהם הבאגים. הייתי על המשמר כדי להציע תמיכה אם וכאשר ביקשו, בנוסף עברתי על כל הבאגים כדי להיות מעודכן. במהלך כל המפגשים שני הצוותים פתחו כמות גדולה מאוד של באגים ורק אני בשל היותי בודק התוכנה המוסמך היחיד הייתי יכול לסקור אותם. הסקירה כללה בדיקה כי אכן מדובר בבאג והדיווח היה עקבי לפי המדריך שחילקתי "איך לפתוח באג", היה חשוב לעבור על כל הבאגים במהירות האפשרית כי היו לא מעט באגים משוכפלים. בחלק הראשון של הבדיקות היו לא מעט באגים שנדחו אבל כמו ציידים אמיתיים הם החלו ללמוד מן הטעויות שלהם.

בסוף היום ניהלנו תחקיר. מטרת התחקיר הייתה להודות לכל המשתתפים על השתתפותם, לסכם את בדיקות היום ולרוץ דרך הפרסים. הפרסים הוענקו על פי 2 קריטריונים:

- כמות הבאגים שנמצאו (ואושרו): לא מפתיע שהפרס הזה הלך לצוות עם המפתח הבכיר אשר אחראי על רוב עבודות הפיתוח.
- הבאג הטוב ביותר: זו הייתה דעה סובייקטיבית שלי ושל מנכ"ל החברה אשר הוענק בסגנון האוסקר עבור "Best Bug" לאחר מכן סיימנו את היום עם בירה, פיצה וסרט במשרד.





אייל זילברמן

מייסד חברת **Qualitest**, חברת הבדיקות הגדולה בעולם. בעל ניסיון כבודק, מנהל בדיקות ויועץ לארגונים בתחום הבדיקות. פרסם עשרות מאמרים מקצועיים ונאם בכנסים ישראליים ובינלאומיים. בין המקימים של מגזין בדיקות התוכנה הישראלי הראשון "חושבים בדיקות" וחבר בצוות ה-AB של ITCB® - ארגון ההסמכה הישראלי לבדיקות. מייסד ושותף במספר חברות היי-טק בתחום הבינה המלאכותית ובינה עסקית.

למה צריך בודקים? (דרך חשיבה של בודקים ומפתחים)

רקע

מקצוע הבדיקות הינו מקצוע "צעיר" יחסית. בזמנים בהם החל פיתוח תוכנה (קצת אחרי סיום מלחמת העולם השנייה) ועד לפני מספר עשורים המקצוע לא היה קיים כלל ורוב פעילויות הבדיקות התמקדו בניפוי (Debugging) של קוד התוכנה. למעשה כיום אנו חוגגים 40 שנה למקצוע הבדיקות, אשר על פי רבים החל בספר "האומנות של בדיקות התוכנה" (*The Art of Software Testing*) מאת גלפורד מאיירס אשר לראשונה הפריד את פעילות הניפוי מפעילות הבדיקות והכין את הבמה לבדיקות "מודרניות".

אז למה לא לתת למפתחים לבדוק?

ברוב המקרים בודקים ומפתחים חושבים בצורה שונה. המטרה העיקרית של הפיתוח היא לעצב ולבנות את המוצר. מטרת הבדיקות כוללות אימות ותיקוף המוצר ואיתור תקלות לפני שחרור המוצר. אלו מטרות שונות שדורשות דרך חשיבה שונה. פיצול תפקידי המפתח והבודק יעזור להשיג רמה גבוהה של איכות המוצר.

דרך חשיבה משליכה על ההנחות שהאדם עושה ועל השיטות המועדפות עליו בעת קבלת החלטות ופתרון בעיות. דרך החשיבה של בודק צריכה לכלול סקרנות, פסימיזם מקצועי, מבט ביקורתי, שימת לב לפרטים ומוטיבציה לתקשורת ויחסים טובים וחיוניים. דרך החשיבה של בודק נוטה להתפתח ולהתבגר ככל שהבודק צובר ניסיון.

דרך החשיבה של מפתח כוללת כמה גורמים של דרך חשיבתו של בודק, אבל פעמים רבות מפתחים מצליחים מעוניינים יותר לעצב ולבנות פתרונות, מאשר לחשוב על מה עלול להשתבש עם הפתרונות הללו, מה שגורם להם עיוורון קוגניטיבי כלפי פגמים ותקלות במוצר.

בדיקות עצמאיות (Independent Testing)

רמת עצמאות הבדיקות נחלקת ל-5 רמות שונות:

- אין בודקים עצמאיים - הבדיקות היחידות הנעשות הן אלה שבהן המפתחים בודקים את הקוד של עצמם
 - מפתחים עצמאיים או בודקים עצמאיים בתוך צוות הבדיקות או צוות הפרויקט; יתכן והיו אלה מפתחים שבודקים את התוצרים של עמיתיהם
 - צוות בדיקות או קבוצת בדיקות עצמאיים בתוך הארגון, מדווחים להנהלת הפרויקט או להנהלה הבכירה
 - צוות בדיקות עצמאי מתוך הארגון או מקהילת המשתמשים, או עם התמחות בסוג מסוים של בדיקות כדוגמה שימושיות, ביצועים, תאימות לתקינה, או ניידות תוכנה
 - בודקים עצמאיים חיצוניים לארגון, כאלה שעובדים באתר הארגון (onsite, insourcing) או באתר חיצוני (מיקור חוץ, outsourcing)
- היתרונות של בדיקות עצמאיות מגיע לידי ביטוי במספר היבטים:

1. **דרך חשיבה** - בודקים עצמאיים עשויים לזהות סוגי כשלים שונים בהשוואה למפתחים מאחר ויש להם רקע שונה, נקודת מבט טכנית שונה, והטיית שונות. בודק עצמאי יכול לוודא, לאתגר או להפריך הנחות שנעשו על ידי המפתחים והמאפיינים של המערכת
2. **התמקצעות** - בודקים עצמאיים מתמקצעים ומגיעים לרמת טכניקה והתמחות גבוהה יותר. זה ההבדל בין שיפוצניק כללי לבין צבעי מקצועי – כנראה שצבעי יידע לצבוע את הבית שלכם טוב יותר בגלל ההתמחות וההתמקדות בתחום
3. **הבנת הלקוח** - איכות אינה תפיסה סובייקטיבית של המשתמשים והלקוחות על איכות המוצר. לפיכך, הגעה לאיכות כוללת הבנה טובה יותר של צרכי וציפיות המשתמשים. לבודקים עצמאיים יש יכולת טובה יותר להבין את ההיבטים העסקיים והתפעוליים של הארגון ולפיכך מבינים טוב יותר את דרך החשיבה של המשתמשים
4. **דיווח לא מוטע** - צוות בדיקות עצמאי המדווח להנהלה יכול לדווח על תוצאות הבדיקות בצורה הוגנת וללא חשש מניגוד עניינים
5. **מוטיבציה** - רוב המפתחים לא אוהבים לעשות בדיקות. רוב האנשים לא טובים בלעשות דברים שהם לא אוהבים. המסקנה – עדיף להביא בודקים שאוהבים לבדוק

סיכום

על אף שמקצוע הבדיקות הינו מקצוע חדש יחסית, הוא מהענפים המתפתחים בקצב המהיר ביותר מבין מקצועות המחשוב ובכלל. עבודת הבודק דורשת דרך חשיבה שונה מפיתוח ויש חשיבות רבה ויתרונות רבים לעצמאות צוות הבדיקות. להיות בודק תוכנה זה לא מקצוע, אלא דרך חיים. הרבה בודקים מנוסים (כולל כותב שורות אלו) מעידים על עצמם כאנשים סקרנים אך באותה מידה גם ספקנים, ולא רק במשימות העבודה אלא גם בחיים האישיים.

להיות בודק תוכנה זה לא מקצוע, אלא דרך חיים





תקציר

"למה זה לוקח כל כך הרבה זמן?", "אנחנו מבזבזים זמן!", "מתי אקבל את הדוח?", "המבדקים שלנו אוטומטיים, למה הם לא רצים?"

כולנו מכירים את המשפטים האלה - אנחנו מפתחים או מטמיעים סביבה, כותבים מבדקים אוטומטיים, בונים מעבדה לתפארת, ועדיין לוקח לצוותי הרגרסה שלנו יותר מדי זמן עד שהם שולחים תוצאות.

הסביבה בה עבדנו מורכבת ועמוסה מאוד. היא מכילה עשרות עמדות עם מאות מחשבים ניידים, מחשבים שולחניים ומכונות וירטואליות והמוצר אותו אנו בודקים הוא כרטיס ה-WiFi BT, המפותח באינטל. המבדקים רצים כבדיקות מערכת וברמת מערכת ההפעלה, וכוללים בדיקות פונקציונליות, בדיקות ביצועים וצריכת הספק, בדיקות UX, עומסים, יציבות, benchmarking ועוד. בסביבת ה-Staging הפנימית שלנו, בה נבדק המוצר מבחינה חומרתית ביחד עם גרסת תוכנה עדכנית, אנו מריצים אלפי מבדקים בכל שבוע.

כאשר התחלנו לנתח לעומק את התפלגות התוצאות גילינו מספר ממצאים: בעוד שתוצאות "עבר" או "נכשל" הן תוצאות רצויות- המוצר התנהג כמצופה או התנהג שלא כמצופה - מה לעשות עם מבדקים שהסתיימו שמקורה בתקלה במערכת האוטומציה? בעיות במחשב הנייד או בסביבה שמנעו מהמבדק להתבצע? או אפילו מבדקים שלא התבצעו לחלוטין על אף שהיו אמורים?

אנחנו לא יכולים להסיק מהמבדקים האלו שום דבר בנוגע לאיכות המוצר שלנו. יותר מזה, כל מבדק שמסתיים ב-"Error", "Blocked", "Not Run" (או בקיצור NRF - Not Run Family), צורך מאיתנו משאבים נוספים - עבודה נוספת של איש הצוות שחוקר אותם וזמן הרצה נוסף ש"תופס" את המעבדה על חשבון דברים אחרים שניתן לעשות בעמדה. לכן שמנו לנו למטרה להוריד את כמות המבדקים שמסתיימים באופן הזה למינימום.

כדי לעשות זאת ולשפר את הביצועים שלנו, פיתחנו את התהליך "Initial Results Analysis". הוכחת היתכנות של הרעיון ביצענו במהלך יולי-נובמבר 2016. זה המקום לציין ולהודות למנהל שלי דאז, דור רוזנברג, על התמיכה והיד הכמעט חופשית שנתן לי לרכז ולהוביל את הפעילות כפי שראיתי לנכון. לאחר הערכה של התוצאות, לקיחת מספר נקודות לתיקון, כמה שינויים שעשינו בצורת העבודה, בסביבה ובכלים בהם השתמשנו, הגענו למודל סופי שנכנס כתהליך למידה ושיפור אינטגרטיבי בסביבה שלנו החל מספטמבר 2017. תהליך המתרחש גם כיום.

דרכים ואמצעים

סביבת הבדיקות שלנו היא "Lights off lab", מעבדה אוטומטית ב-100%. כלומר, כל התהליך מתרחש ללא צורך בהתערבות אנושית. החל בהתקנה של גרסה עדכנית של הסביבה עצמה, כל תהליכי ההתקנה על המכונה לרבות התקנת מערכת ההפעלה, התקנת הגרסה הנבדקת במחשבים הניידים כולל אמצעי debug, הרצת המבדקים והרצה חוזרת במקרים בהם מבדק נכשל (על מנת לקבוע reproduction rate) וכלה בשליחת דוח מעודכן ומפורט הכולל את כל המידע הרלוונטי. הכל נעשה באופן אוטומטי לחלוטין.

צוות הבדיקות שלנו, שאחראי על תפעול הסביבה, נדרש לדווח ולתת מידע על מבדקים שנכשלו בשל בעיות מוצר, וכעת הוא אחראי לתת



אורי קופפרשמיד

בעל תואר B.Sc. בהנדסת חשמל ומחשבים, 13 שנים בתחום הבדיקות באירגון ה-WiFi באינטל.

הוביל מספר תחומים בבדיקות כמוביל טכני וכמנהל צוות. במשך 4 שנים לקח חלק בצוות סביבת ה-Staging האוטומטית באירגון, ועוסק כיום בתחום הסייבר בטכנולוגיות WiFi ו-Bluetooth.

מטרות והנחות

המדד הראשון שהגדרנו הוא NRF % - מספר המבדקים מסוג NRF לחלק לסך המבדקים-

$$NRF\% = \frac{Total\#of\ "Not\ Run" + Total\#of\ "Error" + Total\#of\ "Blocked"}{Total\#of\ test}$$

האנליזה הראשונית שלנו הראתה כי למעלה מ-30% מהמבדקים שלנו הם NRF. זה המון. מתוך מטרה להוריד את הזמן שמתבזבז בגלל ה-NRF, הצבנו יעד אגרסיבי של הורדת NRF ב-80% מ-30% ל-5%.

ניתן לחלק את הבעיות למספר קטגוריות:

- בעיות יציבות של המכונות / המחשבים
- בעיות באוטומציה של המבדקים
- בעיות בסביבה / תשתית
- טעויות אנוש / אינטראקציה שגויה עם המערכת
- תקלות בצידוד היקפי כגון ראטרים, נתבים וכו'
- שונות: בעיות כלליות שאינן מתאימות לקטגוריות הקודמות

באופן סטטיסטי, בחרנו להתמקד במחזור בדיקות שבועי סטנדרטי של כמה מאות בדיקות שמייצגות היטב את אלפי הבדיקות שאנו מריצים כל שבוע. אנליזה ראשונית התבצעה ביום ראשון בשעות הבוקר, אחרי שהמבדקים רצו במהלך סוף השבוע.

שמנו לב שעצם העובדה שהתחלנו לדבר על העניין וקבענו ישיבה שבועית קצרה עם הצוות שמרין, אפשרה לנו לראות את התמונה הכוללת של המצב. שיפרנו את התקשורת בתוך הצוות ובאופן מיידי ראינו שיפור בתוצאות. הסיבה לכך היא העובדה שחברי הצוות שמו דגש יותר משמעותי על משמעת ובקרה אישית וניסו להימנע מטעויות (נושא זה מכוסה בהמשך בהרחבה). הדבר מתיישב עם מחקרים שנערכו בתחום, ועם Hawthorne Effect שהדגים כי אנשים משנים את התנהגותם כאשר הם מודעים לעובדה כי בוחנים את התנהגותם. ההחלטה הייתה להגדיל את תכיפות הפגישות לפעמיים בשבוע. ישיבה ראשונה ביום ראשון בה נגענו בפרוץ בבעיות הגדולות שצצו בעמדות השונות ובישיבה השנייה ביום רביעי לאחר שהכל נגמר ויש בידנו יותר מידע, "חפרנו לעומק" בכל הבעיות.

מדד נוסף שהגדרנו היה מדד עזר השוואתי TTI (Test to Issues). הוא מוגדר כסך המבדקים ממשפחת NRF, לחלק לסך סוגי הבעיות שזיהינו. ערך TTI גבוה משמעו כי באופן ממוצע כל בעיה השפיעה על הרבה מבדקים, בהשוואה לערך TTI נמוך.

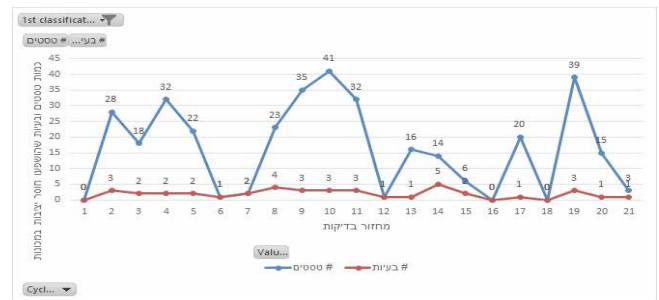
תמיד נאתגר את עצמנו כדי להשיג סביבה ואוטומציה יותר יעילה ואפקטיבית



בעיות יציבות של המכונות / המחשבים

כיוון ששיפור יציבות המכונות הוא כמעט ובלתי אפשרי, התמקדנו באיסוף נתונים, הגדלת הידע שברשותנו ומציאת דרכים לצמצום ההשפעה של בעיות כאלה.

מהנדס מומחה לנושאי יציבות פלטפורמות / מחשבים בפיתוח הצטרף לשישיה ביום ראשון ובחן את הבעיות ביחד איתנו. סיוע ושיתוף הפעולה הגדיל את הידע שלנו בנושא באופן דרמטי. למשל, מהו ההתהליך הנכון לאיסוף מידע ממכונה שקרסה ומה המידע אותו יש לאסוף כדי לאפשר חקירה מוצלחת של בעיית היציבות ע"י הצוותים הרלוונטיים. דבר נוסף שעשינו הוא להחליף בין דגמי מכונות שעדיין בשלבי פיתוח (ולכן פחות יציבות) ובין מכונות שיצאו לשוק באופן מסחרי (ולכן יותר יציבות), כאשר על המכונות היותר יציבות הרצנו ובעלות סבירות גבוהה יותר לגרום שמעמיות על המערכת עצמה ובעלות סבירות גבוהה יותר לגרום לקריסה של המכונה, על הפחות יציבות הרצנו בדיקות שיוצרות פחות עומס על המכונה עצמה וכך הגדלנו את היציבות באופן כללי. אבחנה נוספת שראינו היא שכמות הבעיות שנתקלנו בהן הייתה דומה מדי שבוע. הדבר הגיוני, כי תצורת המכונה (למשל גרסת BIOS) לא משתנה במרווחי זמן קצרים כל כך.



איור 1: כמות מבדקים ובעיות שהושפעו מחוסר יציבות במכונות

ערך מדד TTI-8.7. הערך השני הכי גבוה (ביחס לקטגוריות האחרות), כפי שניתן לראות בעייה אחת יכולה להשפיע על כמות גדולה של מבדקים, בייחוד אם היא מתרחשת בתחילת מחזור הבדיקות. דבר נוסף שראינו הוא שבעיות כאלה קורות כמעט בכל שבוע ואין מוגבלות לעמדה ספציפית.

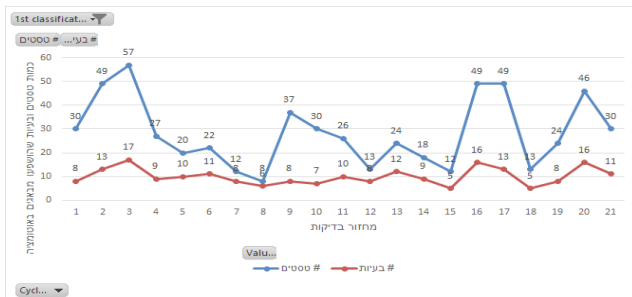
מה לעשות עם מבדקים שהסתיימו בשגיאה שמקורה בתקלה במערכת האוטומציה, בעיות במחשב הנייד או בסביבה שמנעו מהמבדק להתבצע

בעיות באוטומציה של המבדקים

דבר מרכזי שהבנו הוא שבעיות באוטומציה נוטות לחזור על עצמן באותן עמדות ולאורך זמן. עבדנו יחד עם הצוות שכתב את המבדקים האוטומטיים והגדרנו את דרך העבודה הנכונה. החלטנו להשתמש בפרויקט קיים ב-JIRA לצורך מעקב אחרי באגים של האוטומציה והגדרנו Bug Life Cycle עדכני. קבענו ישיבת Bug Scrub שבועית עם צוות הפיתוח ולקוחות נוספים של האוטומציה, ישיבה בה עקבו אחרי הבאגים שנפתחו בשבוע החולף, דנו בחשיבות ובדחיפות של הבאגים הידועים וטיפלנו בבעיית ידע ובבקשות למידע נוסף שהתקבלו מצוות הפיתוח. הגדרנו מחדש לוחות זמנים ועקרונות עבור שחרור של גרסאות אוטומציה. זיהינו את נקודות התורפה בשלב ההיתכנות ובנינו עמדות ייעודיות לבדיקת יציבות המבדקים עצמם.

למעשה שינינו את הצורה בה אנחנו מתייחסים לאוטומציה. זה כבר לא אוסף סקריפטים או פרויקט משותף של כמה אנשי אוטומציה, אלא מוצר תוכנה לכל דבר ועניין ולכן יש להתייחס אליו כמוצר תוכנה - לוחות זמנים, בעלי תפקידים מוגדרים (מנהל מוצר,

מנהל פיתוח, מנהל בדיקות וכו'), מדדים אובייקטיביים של איכות ויציבות ומתודולוגית פיתוח תוכנה (במקרה שלנו - Continuous Integration). ההישג המשמעותי ביותר היה הקמה של עמדות Pre Checkin, עמדות שמטרתן היא מציאת באגים ושגיאות יסודיות בקוד האוטומציה לפני שקוד בדיקה חדש נכנס.



איור 2: כמות מבדקים ובעיות שהושפעו מבאגים באוטומציה

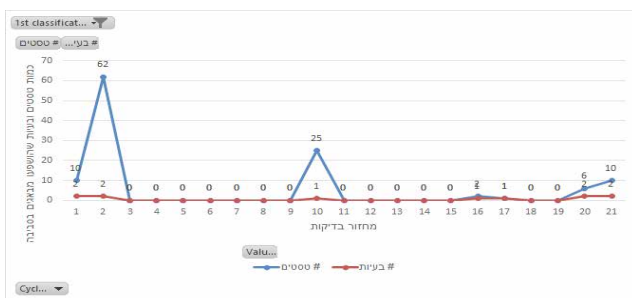
ערך מדד TTI-2.8, הערך השני הכי נמוך שמדדנו. משמעות הדבר היא כי באופן יחסי יש מעט מבדקים בעייתיים לכל באג.

עליות חדות בגרף (למשל במחזורי הבדיקות 3, 13, 16 ו-20) הם ערכים שמדדנו לאחר קבלת עידכון של גרסת האוטומציה. הדבר מדגים היטב את האמירה כי "על כל באג שמתוקן, 2 חדשים נכנסים". לצערנו, נוכחנו לדעת כי הדבר מדויק...

פערים ביכולות התשתית

בשלב הוכחת ההיתכנות זוהה פער משמעותי בתשתית שלנו - לא היה לנו מנגנון להפריד בין התוצאות הראשוניות של המבדקים לתוצאות הסופיות שדווחו למנהלי המוצר.

צוות התשתית סיפק עבורנו מנגנון כזה עם יכולות סינון מוטמעות בתוך הסביבה שאיפשרו לנו איסוף נתונים והתמקדות באזורים שמעניינים אותנו. בנוסף הם פיתחו עבורנו דוח ייעודי וחסכו לנו שעות רבות של עבודה יחסית למה שנעשה בשלב ההיתכנות, והפכו את המנגנון לפשוט ואינטואיטיבי.

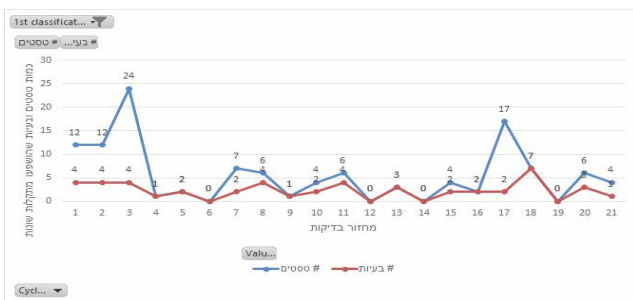


איור 3: כמות מבדקים ובעיות שהושפעו מבאגים בסביבה

ערך מדד TTI-10.5, הערך הגבוה ביותר שמדדנו, וזה הגיוני היות וכל בעיה בתשתית או בסביבה משפיעה על מספר עמדות ועל כמות גדולה יחסית של מבדקים. מצד שני יש לציין כי הזמן לתיקון בעיות תשתית היה מהיר מאוד ובעיות טופלו באופן מיידי.

טעויות אנוש / אינטראקציה שגויה עם המערכת

כפי שהזכרתי, קיימנו 2 ישיבות. כל טעויות האנוש זוהו בישיבה הראשונה - שימוש בדרייבר לא מתאים, הפעלת חבילת מבדקים לא נכונה ועוד. במספר מקרים שמנו לב כי טעויות נוטות לחזור על עצמן ע"י מספר אנשים שונים, אך ללא תהליך בו זיהינו ודיברנו על אותן בעיות, לא יכולנו לטפל בהן. על מנת שזה יקרה ושנוכל



איור 6: כמות מבדקים ובעיות שהושפעו מתקלות שונות

ערך מדד $TTI \sim 2.5$ הערך הנמוך ביותר שמדדנו, וגם זה הגיוני בשל אופיין של הבעיות הנ"ל. הצעדים שנקטנו היו תמיד זהים - להמשיך לנטר את העמדה, ובמידה שהבעיה משתחזרת, יש לסווג אותה לאחת מהקטגוריות האחרות.

מסקנות

מעבר מבדיקות ידניות לאוטומטיות והקמת סביבה מתאימה היא הכרחית אך לא מספיקה. אוטומציה היא אמצעי להשגת מטרה ואנו צריכים לשאוף תמיד למקסם את השימוש ונצילות העמדות שלנו.

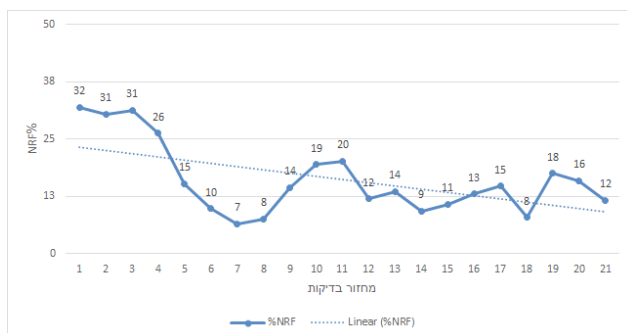
כפי שהדגמתי, העולם אינו מושלם. עדיין יש לנו בעיות עם מוצר התוכנה הפנימי (הידוע בשם הגנרי "אוטומציה"), עדיין יש בעיות בציוד ובמכונות ואנחנו עדיין עושים טעויות. אף על פי כן, תמיד נאתגר את עצמנו כדי להשיג סביבה ואוטומציה יותר יעילה ואפקטיבית.

בשלב הסופי של התהליך יש לשאול מה השגנו מתוך היעדים שהצבנו:

- ערכי $NRF\%$ גבוהים של כ-30% לא נראו יותר לעולם
- לא הגענו עד ליעד של 5%
- הערך הנמוך ביותר שהצלחנו להשיג היה 7%
- קו המגמה יורד והגיע לכ-10% בסוף הגרף

כפי שתיארתי, תהליך ומתודה זו תוכננו עבור סביבה מאסיבית בהיקפה. מימוש של תהליך זה משמעותו הקצאת משאבים וזמן לצורך למידה ופתרון בעיות והורדה של $NRF\%$. הדבר איפשר לנו להגדיל את קיבולת הסביבה שלנו, להוריד את הזמן עד לשליחת הדוח הסופי ב-50% (יומיים) והשפעה חיובית ישירה על ניצולת העמדות.

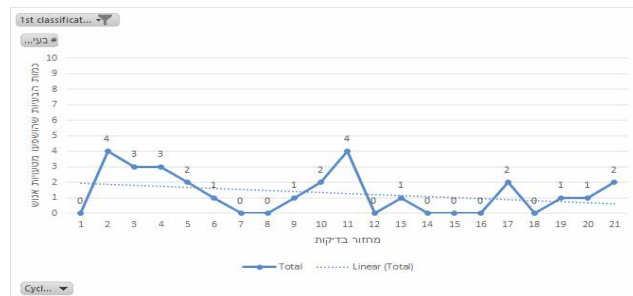
כל קטגוריה טופלה בדרך שונה - החל משיבות פנימיות של הצוות דרך Bug Scrub עם שותפים וכלה בתקשורת והעברת ידע בין קבוצות אחרות שאיפשרו לנו לפעול ולשפר את עצמנו.



איור 7: $NRF\%$ לאורך הזמן

ללמוד מטעויות עבר, התחלנו לנטר את הבעיות שאנחנו גרמנו.

פתחנו פרויקט חדש ב-JIRA, שהיה זמין רק לצוות שלנו, בו פתחנו "Tickets" על הטעויות והשגיאות שלנו. אדגיש - לא חיפשנו "אשמים", לא הפנינו אצבעות ולא התעמרנו במהנדסים שעשו טעויות. "Cuiusvis hominis est errare" לטעות זה אנושי, אמר מרקוס טוליוס קיקרו [106-43 לפנה"ס], כלנו בני אדם ואנחנו רוצים ללמוד מהטעויות שעשינו כדי להשתפר לפעם הבאה.



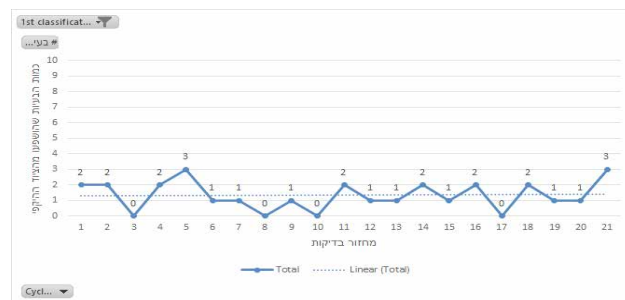
איור 4: כמות הבעיות שהושפעו מטעויות אנוש

מהתבוננות במספרים, ניתן לראות מגמת ירידה בכ-50%, אך חשובות יותר הן 2 האבחנות הבאות:

1. אף טעות לא חזרה על עצמה אצל אותו אדם
 2. המהנדסים בצוות הפנימי את התהליך ועיקרון הלמידה והיו פתוחים לדבר ולדון בטעויות שהם עצמם עשו
- ערך מדד $TTI \sim 6.8$ יחסית גבוה, דבר המצביע על שגיאה רוחבית שעלולה להשפיע על מס' רב של מבדקים.

תקלות בציוד היקפי כגון ראוטרים, נתבים וכו'

תקלות בציוד ההיקפי הן למעשה בעיות בציוד - נתבים, מתגים, נפח פנוי בדיסק ועוד.



איור 5: כמות הבעיות שהושפעו מהציוד ההיקפי

ממוצע הבעיות לאורך הזמן הוא 1.3 והוא ככל הנראה הערך הנמוך ביותר אליו ניתן להגיע, בהתחשב בגודל הסביבה שלנו ובכמות המבדקים. בעיות בציוד טופלו באופן מיידי, לרוב ע"י החלפת הציוד. אף לא בעיה אחת נצפתה פעמיים על אותה עמדה.

ערך מדד $TTI \sim 7.7$ יחסית גבוה, שוב, דבר המצביע על שגיאה רוחבית שעלולה להשפיע על מס' רב של מבדקים.

שונות / בעיות כלליות שאינן ניתנות לסיווג

כפי שמשמע מהשם, אלו בעיות נדירות ולא סדירות שאין לנו מספיק נתונים עליהן כדי לפעול למניעתן בעתיד.



אל תחזור על עצמך

DRY – אל תחזור על עצמך מוכר זה מהנדסת תוכנה מקבל פנים רבות כשאנו מסתכלים מנקודת מבט של הבודקים. במקור, בא מונח זה להזכיר למתכנתים ואנשי אוטומציה בתוכם, כי לא רצוי לחזור ולכתוב מחדש שוב ושוב קטעי קוד הבאים לטפל באותה מטרה - אלא כאשר נזהה כי הפעולה אליה אנו מכוונים כבר התבצעה במקום אחר בתוכנה - עדיף להוציא פעולה זו כפונקציה גנרית לה נקרא ממקומות שונים בתוכנה (בכל פעם בה נצטרך לבצע פעולה גנרית זו, עם אותם נתונים או נתונים אחרים). בכך נצמצם את גודל התוכנה ובו בזמן נקל על עבודת התחזוקה, כלומר, כאשר זיהינו בעיה באלגוריתם - כעת נוכל לתקן במקום אחד מרכזי במקום לחפש ולתקן עוד ועוד במקומות דומים.

הגיוון שבשילוב שיטות שונות ומציאת האיזון ביניהן, היא שמביאה לעבודה מושלמת ככל הניתן ולכיסוי מיטבי בבדיקות.

כאשר מסתכלים מנקודת מבט של עבודת הבודקים - גם ניתן לזהות מקומות בהם רצוי כי לא נחזור על עצמנו:

1. היתרונות בשימוש בסטים מגוונים של ערכים - לעומת חזרה חוזרת ונשנית על אותה הבדיקה עם אותו סט ערכים בדיוק. דבר המביא להגדלת הסיכוי למציאת באגים חדשים לעומת תופעת ההתחסנות של התוכנה Pesticide Paradox (מזכירה התחסנות החרקים הנותרים מפני תרסיסי רעל בהם נעשה שימוש חוזר ונשנה).
2. בצורה דומה נרצה לגוון בשימוש בנתיבים שונים בתוכנה המביאים לאותה התוצאה. לדוגמה: שימוש מדי פעם בתפריט הימני לעומת שימוש קבוע בתפריט הראשי, או ביצוע פעולות לכאורה בלתי תלויות, בסדר שונה בכל חזרה על אותו מקרה הבדיקה. לעיתים מה שחשבנו כי יהיה בלתי תלוי - מתגלה לפתע כבעל השפעה שלילית על התנהגות התוכנה.
3. גיוון בין עבודה ידנית לאוטומציה - כתיבת מסמכים מפורטת לשימוש בהנחיות על ורשימת רעיונות, יציאה מתכנית העבודה המוגדרת לחקירת רעיונות שעולים לנו לפתע, שילוב מידה נכונה של בדיקות חקרניות (Exploratory Testing) כל אלו מוכיחים את יתרונם בכך שאין שיטה אחת הטובה מכולן, אלא דווקא הגיוון שבשילוב שיטות שונות ומציאת האיזון ביניהן היא שמביאה לעבודה מושלמת ככל הניתן ולכיסוי מיטבי בבדיקות.
4. רבים נוטים לשכוח כי גם בכתיבת מסמכים רצוי לזכור את עקרון ה-DRY ניתן להפנות ממקומות רבים לתהליכים כתובים במקום מרכזי, ולחסוך זמן כתיבה רב בכך שהוראות הביצוע המפורטות יוחזקו במסמכי How-To או יקושרו בכלל למסמכי העזרה וההנחיות למשתמש - User Manuals, Help windows והנחיות תפעול שונות (כך אנו מרוויחים רווח כפול כאשר יותר בודקים עוברים על מסמכים אלו ומשפרים גם אותם).

כולנו נוטים להשתמש בהרגלים קבועים ולכן חשוב מאוד להחליף תפקידים



5. חשוב לזכור כי גם מסמכי הבדיקות חייבים להיות כתובים כך שישאירו שיקול דעת לבודקים המבצעים את הבדיקות בפועל, וינחו אותם כיצד לגוון, וכיצד להמשיך ולהסתכל מסביב ולחשוב על דרכים אחרות לביצוע שאולי נשכחו ע"י הכותב המקורי.

6. כולנו נוטים להשתמש בהרגלים קבועים (גם בעצם הצורה בה אנו מגוונים) ולכן חשוב מאוד להחליף תפקידים ומדי פעם לתת לבודקים אחרים לבצע בדיקת נושא מסויים. אמנם מי שבדק אותו בעבר כבר מנוסה ובוודאי יבצע זאת באופן מהיר יותר, אך בהחלט סביר להניח כי בודק אחר יעלה שאלות חדשות ודרכים מעט שונות לביצוע כמו גם יתמקד בנקודות שונות אשר אולי ניתן להן פחות דגש בעבר ("אורח לרגע רואה כל פגע").
7. למנהלים ולראשי הצוותים נציע - גוון את הרקע ושנות הניסיון של הבודקים אותם אתם מגייסים - אמנם נכון כי קל יותר לנהל אנשים החושבים בדיוק כמוכם, אך היתרונות בהחזקת צוות מגוון יביא לרעיונות ושיפורים ניכרים בשיטות העבודה.
8. גם בהרצה אוטומטית - רצוי להקפיד ולגוון את סדר הריצה בין הבדיקות אחת למספר סבבי ריצה, כי קיימת סבירות גבוהה כי גם להיסטוריה שעברה על המערכת לפני ביצוע בדיקה מסוימת, יש השפעות על התנהלות הבדיקה והמערכת. אם אנו נוהגים תמיד לנקות המערכת לפני ריצה, ייתכן מאוד והריצה תיכשל אם נותרו שאריות של ריצות קודמות, חלק מהזיכרון כבר נאכל, יש פחות מקום בדיסק ועוד.

חומר קריאה נוסף

<https://www.mkltesthead.com/2013/08/testers-dont-repeat-yourself-99-ways.html>

נשמח לשמוע רעיונות הערות והארות מכם הקוראים - בפורום בקב' הפייסבוק: <http://bit.ly/TestIL-FB>

אתם מוזמנים לקרוא טיפים רבים נוספים ולהוסיף טיפים משלכם לרשימה משותפת זו. <http://goo.gl/UcW42>

סדרת טיפים זו "כיצד להפוך לבודקים טובים יותר" מתבססת על דיון ב: Software Testing Club

[99 Things Testers Can Do To Become Better Testers](#)

ה-eBook החינמי שנוצר בעקבות דיון זה: [99ThingsEbook.pdf](#)

וסדרת פוסטים מאת Michael Larsen בשם: [Ways Workshop 99](#) - בה מיכאל מרחיב על כל אייטם וגם מספק הנחיות כיצד לתרגל הנושא.

9. ברוב המערכות קיימים משתמשים רבים העובדים במקביל ולא פעם נראה כי הבאגים הקשים לאיתור מוקדם קשורים במגוון הפעולות הרצות ברקע במקביל לבדיקה שאנו מבצעים.

לעיתים כמובן שנצטרך גם לחזור על אותן הבדיקות בדיוק, במיוחד כאשר אנו מכירים סטים של ערכים החשובים יותר ללקוחותינו, או תהליכים נפוצים הנוטים להישבר יותר מאחרים, אך פרט לכך -

מסתבר כי שינוי הרגלי העבודה הנו חשוב מאוד להשגת תוצאות מרביות במהלך הבדיקות.



קובי הלפרין - עוסק בבדיקות מזה מעל ל-20 שנה בעיקר בתחום התקשורת טלפון/דטאקום (ECI, Alvarion), בעל מעורבות גבוהה בקהילות בעולם ובארץ ומוביל מספר פורומים אינטרנטיים, מחפש להרחיב את הדיון בתחום הבדיקות, שיפור כלים והרחבת הידע של הבודקים. פעיל בוועד המייעץ של ITCB® - בעיקר בשיפור האתר www.itcb.org.il והקשרים ברשתות החברתיות.





פרסים יקרי ערך הוענקו לזוכים בגמר

מקום I - טיסה, אירוח והשתתפות בכנס Agile Testing Days 2019 שיתקיים בפוסטדאם, גרמניה בנובמבר 2019.

מקום II - קורס מקצועי (עד 40 שעות) וטאבלט בחסות מכלל גו'ן ברייס

מקום III - קורס מקצועי (עד 40 שעות) בחסות מכללת גו'ן ברייס

העולים לגמר זכו בכרטיס חינוך

לכנס QAWeek 2019

מזל טוב לאלופי הבדיקות של ישראל

זו השנה השלישית לתחרות הבדיקות הישראלית ISTC שהתקיימה בזכות יוזמתה של אנה לוקובסקי ובתמיכתם של ITCB - העמותה הישראלית להסמכת בודקים, Matrix Testing & Automation, ג'ון ברייס הדרכה, HomeDiet & Gamitee - the SUT owners, Inflectra, Agile Testing Days 2019-1.

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

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

במוקדמות שנערכו ב-5 לאפריל 2019 השתתפו 17 צוותים ודווחו 340 תקלות.

בגמר שנערך ב-24 ליוני 2019 השתתפו 5 צוותים ונמצאו 70 תקלות.

צוות השופטים במוקדמות ובגמר כלל מנהלים ויועצי בדיקות בכירים המייצגים את חברות ה-IT וההייטק המובילות בישראל- אנה לוקובסקי (יו"ר ISTC וצוות השופטים), ירון צוברי (נשיא ITCB), דני אלמוג (מרצה לנושאי איכות תוכנה במכללת סמי שמעון), דדי תנעמי (מנהל מדור QA בכאל) ומשה מאמיה (מנהל QA ב-MicroFocus).

הזוכים של ISTC 2019

מקום ראשון

קובי צם, מפתח iOS מחברת תן ביס ו סיגל שעשוע, מנהלת QA

מחברת חשבונית ירוקה

"אנחנו זוג נשוי מרמת גן, הורים לילד חמוד. הגענו לתחרות בשביל האתגר, ברגע ששמענו על התחרות, שקלנו לגשת גם כדי לבדוק את היכולות שלנו, וגם כחוויה זוגית."

האתגר הראשון היה מקצה המוקדמות.

היות ואף אחד מאתנו לא השתתף מעולם בתחרות כזאת, לא בדיוק ידענו לקראת מה אנחנו הולכים. בסופו של דבר לא היינו כל כך מרוצים מהדו"ח שהגשנו.

עם זאת, למדנו מהתהליך והסקנו מסקנות לגבי חלוקת זמן טובה יותר וחלוקה נכונה של העבודה בינינו.

כשהגענו למקצה הגמר כבר ידענו בדיוק מה צריך לשפר ולכן העבודה זרמה בצורה חלקה ויעילה יותר, מה שאפשר עבודת בדיקות איכותית (ומהנה) יותר והגשנו דו"ח שהיינו הרבה יותר שלמים איתו מזה של המקצה הראשון.

אם נסכם ניתן לומר שמאוד נהנינו מהתהליך, משיתוף הפעולה בינינו, מהלמידה, מהאנשים וכמובן מהזכייה.

אנחנו רוצים להודות לאנשי ISTC על התהליך ועל החוויה הייחודית."

מקום שני

מיכל ראמי ראש צוות בדיקות וניצן שוקר מהנדס בדיקות מחברת Mmuze

"אנו מובילים את תהליכי ה-QA בחברת Mmuze ועובדים ביחד מזה כשנתיים. החלטנו להשתתף ביחד בתחרות מכיוון שראינו בזה אתגר."

סברנו כי אנחנו יכולים להעתיק את הכימיה הטובה והתהליכים שפיתחנו ביחד בצוות גם לתחרות. שנינו אנשים סקרנים ששואפים להשתפר ולהעשיר את הידע והיכולות שלנו. כך שפרט לאתגר התחרותי אנו רואים בזכייה הזדמנות לרכוש ידע ויכולות נוספים שימשיכו לעזור לנו להתפתח בתחום."



מקום שלישי



איתמר משה מהנדס מערכת ור"צ בדיקות וסופי דנילוב בודקת תוכנה מחברת Puzzle Projects

"אנחנו עובדים ביחד בצוות בדיקות מספר שנים. בתקופה זו, התמקצענו בתחום בדיקות התוכנה, הבאנו ליציבות ותקינות מערכות רבות. כצוות אנו שמים דגש רב על עבודה משותפת עם צוות הפיתוח וצוות הנדסת המערכת. האינטראקציה איתם חשובה מאוד על מנת להבטיח את איכות המוצר אותו אנו מספקים. המקצועיות אותה אנו דורשים על מנת לשמור על איכות המוצר, אינה באה בחשבון על האווירה בה הצוות עובד. הצוות עובד גם למהנה. היתרון של הצוות שלנו הוא, שכל אחד לומד וצובר ידע מהחבר האחר בצוות."

החלטנו להשתתף ב-ISTC מכיוון שהיינו סקרניים להכיר את תהליכי הבדיקות בחברות אחרות, אנו רואים בזה אתגר חדש עבורנו. רצינו לבחון את הניסיון והידע שלנו מחוץ למקום העבודה המוכר והבטוח. אנחנו מאוד שמחים שעברנו את חלק הראשון של התחרות! היה מעניין וכיף! מחכים לשלב הבא!

- אנחנו מאמינים במספר דברים:
- תמיד יש באגים! אין מערכת עם 0 באגים.
- בודק תוכנה טוב אמור להכיר את המערכת יותר טוב מכולם (גם ממפתחים)
- It's a BUG and NOT a FEATURE
- מערכת אף פעם לא עובדת כמו שצריך כאשר מפתח אומר – "אבל אצלי במחשב זה עובד טוב..."

ברכות גם לשאר הצוותים שעלו והתחרו בגמר גביע זכו בתארים מיוחדים

Best Enhancement Award



יובל מנושה, QA Analyst & QA Guild Manager מחברת Via

"בחרתי להשתתף בתחרות כי אני אוהב אתגרים. רציתי לראות כיצד אתפקד מול צוותי בדיקות מהטובים בתחום, ועל הדרך לנסות ללמוד כמה דברים חדשים מהמתמודדים האחרים."

ה"אני מאמין" שלי הוא תמיד ללמוד, להשתפר ולאתגר את עצמך ולתת לסקרנות להוביל אותך. בתחום הטכנולוגי שלנו, שתמיד מתחדש במהירות גבוהה, אתה תמיד צריך לאתגר את עצמך כדי להישאר בטופ. ואני מאמין שזה התפקיד שלנו לעשות הכל כדי שהמשתמש יקבל את המוצר הטוב ביותר שניתן לספק לו."

Best Accessibility Testing Award



יהונתן דיוויס, בודק תוכנה מחברת uTest

"אני מאמין באיכות ומאמין באתגרים וכשיש לי הזדמנות להשתתף באתגר איכותי איך אפשר להגיד לא?"

אני מאמין שטמון המון פוטנציאל לא ממומש בתחום בדיקות התוכנה, הרבה מקומות שהתחום יכול להתפתח אליהם ושנחנו כאנשי מקצוע יכולים להתפתח כל הזמן, אחת הדרכים הטובות להתפתחות היא דרך אתגרים ואני מוצא בזה באופן אישי דרך מעולה למתוח את הגבולות שלי ובו זמנית לתרום לאיכות של מוצרים שנראה בעתיד בשוק."

משתתפים	סטטיסטיקות המוקדמות	סטטיסטיקות הגמר
	17 צוותים, 9 זוגות, 8 צוותי יחיד	7 צוותים, 6 זוגות, 1 צוות של יחיד
המוצר הנבדק	HomeDiet מאגר הדיאטנים והדיאטניות הגדול בישראל	Gamitee By tapping the conversational data created in real-time, Gamitee gives shopping sites unique insights and amazing opportunities to sell more and better
פרק זמן הבדיקה	78 שעות אדם	24 שעות אדם
כמות תקלות שדווחו	340	71
כמות ממוצעת של תקלות שנמצאו לצוות	23	14
מקסימום של תקלות שנמצאו ע"י צוות אחד	61	16
מינימום של תקלות שנמצאו לצוות	10	10



05.03.2019



ניצן גולדנברג

מזה 5 שנים בתחום
בדיקות התוכנה.
מהנדס בדיקות בחברת
SeatGeek, המוביל
הראשי של קבוצת
המיטאפ TestIL, עורך
וכתב במגזין "עולם
הבדיקות", מרצה בקורסים
לבודקי תוכנה וחבר בצוות
המייעץ AB של ITCB®.

בחודש מרץ התקיים במשרדי חברת Samanage מיטאפ בנושא
"Talking Automation - Share how we do Automation"

מרצה עומרי מוגוז

בהרצאה עומרי סיפר על חברת Samanage ואיך הם עושים את הבדיקות האוטומטיות שלהם.
סופר כיצד החברה עבדה לפני עידן ה-AWS ואיך AWS שיפרו להם את הבדיקות.
עומרי הסביר על בעיות שהבודקים וצוות הפיתוח נתקלו לפני המעבר ל-AWS, על האתגרים
והיתרונות לאורך הדרך.

מרצה גונן צור

גונן סיפר בהרצאתו על העבודה בחברת RapidAPI ועל ההחלטות שנלקחו בחדר הישיבות של
החברה בכל הקשור לפיתוח האוטומציה:

- ✓ מדוע החליטו לפתח את האוטומציה מאפס ללא מסגרת
- ✓ מדוע האוטומציה שלהם איננה מושלמת By-design
- ✓ מה הם למדו מעבודה עם Testim.io
- ✓ מהי אסטרטגיית הבדיקה והתפיסה של החברה
- ✓ מה הייתה ההחלטה הגרועה ביותר שהם קיבלו



07.04.2019

בתחילת חודש אפריל התקיים בחברת Astea בכרמיאל מיטאפ בנושא אוטומציה - הצעד הבא לבודקי תוכנה
וההרצאות הועברו על ידי יואב לוינסון מחברת Practis

חלק 1 - הרצאה על אוטומציה באופן כללי

ההרצאה התעסקה במושג "אוטומציה" ומה זה בכלל אוטומציה.

היתרונות והחסרונות באוטומציה, אילו סוגי בדיקות ניתן לעשות, איך בוחרים את השפה איתה כותבים את הקוד לאוטומציה, איך
בוחרים את הכלים הנכונים לעבוד איתם, עלות לעומת תועלת ואיך מתחילים להקים מערך של אוטומציה.

חלק 2 - סדנה למתחילים לכתיבת קוד המיועד לאוטומציה

בסדנה המשתתפים למדו איך לכתוב קוד בשפת VBScript בעזרת ה-NOTE++, בנוסף למדו ותרגלו המשתתפים לבנות תרחיש
אוטומציה בסיסי בעזרת סלניום.

*אתם מוזמנים להיכנס לקישור ולתרגל את התרגילים שניתנו בסדנה **קישור**



22.05.2019

בחודש מאי התקיים במשרדי חברת WeWork בתל אביב מיטאפ בנושא "Low Level Automation Android tools"

מרצה תומר כהן

בהרצאתו תומר דיבר על גישת בדיקות של שכבות האבסטרקציה הנמוכות יותר (low level automation), הציג את הכלי ADB של Android והראה כיצד ניתן לכתוב תרחישים אוטומטיים במקומו.

מצגת

מרצה טליה יורמן

אתגרים בבדיקות אפליקציית מובייל



בדיקות מובייל שונות בכך שהנראות והשימושיות חשובות מאד

יש יותר מקרי בדיקה הנגרמים משינויים בתנאי הסביבה תוך כדי השימוש (קישוריות, קליטה, אור, רעש וכו')

מטריצת המכשירים היא אין סופית, יש לבחור מכשירים + מערכות הפעלה שיהיו תחת בדיקה באופן מושכל

יש צורך לאפיין את פלטפורמות המכשירים הנפוצים ביותר לשימוש ולעקוב אחרי השוק

יש צורך להכיר ולאפיין את אוכלוסיית המשתמשים כדי להתמקד במקרי בדיקה נפוצים

יש לשים לב לתפקוד האפליקציה ב-Settings שונים של המכשיר

כדי להגדיל את הכיסוי מומלץ להשתמש באמולטורים/חווות מכשירים/חברות המפעילות בודקים שונים בכל העולם

גם אחרי שחרור הגרסה, יש לעקוב כל הזמן אחרי השתנות שוק המובייל, ולבצע בדיקות רגרסיה (לפחות) למכשירים/מערכות הפעלה חדשים

מצגת

מרצה גיל זילברפלד

גיל הרצה אודות Dependency Injection אשר הינה טכניקת עיצוב קידוד וזאת בשל היותה הכי נפוצה בתוכנות המודרנית, השימוש בה מאפשר למפתחים ולבודקים לבדוק את הקוד בצורה טובה יותר.

עם זאת, Dependency Injection הנו רק החלטה אחת אשר הארכיטקטורה, עיצוב, קידוד שהמפתחים לוקחים כל יום. כל אחת מההחלטות הללו משפיעה על יכולת הבדיקה, וגם על השימושיות, על תחזוקת האבטחה ועל היבטים רבים של התוכנה. אם כל כך הרבה על הקו, החלטות אלה לא ניתן לנקוט על ידי היזמים, מבלי להבין את ההשפעה. הבודקים צריכים להיות חלק ממחזורי התכנון, תוך התמקדות בשיפור יכולת הבדיקה וביצירת מוצר טוב יותר.

מצגת



היום אני רוצה להציג כמה עובדות ומגמות שישפיעו רבות בעתיד הקרוב על הבדיקות



יאן ברון

מנהל בדיקות בחברת iMDSOFT ישראל.
בעל תואר M.Sc במדעי המחשב
והסמכה בינלאומית של ISTQB® FL
עם 35 שנות ניסיון בהנדסת תוכנה וניהול בדיקות.

תחומי עניין

מתודולוגיה וכלי פיתוח לאזורים שונים בבדיקות, בדיקות אוטומציה, ניהול מחזור חיי מוצר: מתודולוגיה וכלים.

התנדבות

ISTQB® – ראש ועדת סקרים, ITCB®
ועד המנהלים, SiGIST ישראל – יו"ר תכנית הכנסים.

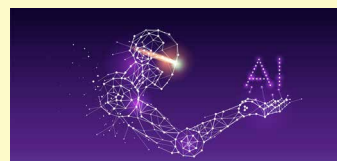
Big Data Statistics 2019

techjury.net, March 22, 2019



- שוק נתוני העתק (big data) צפוי להגיע עד שנת 2023 ל-103 מיליארד דולר
- בשנת 2019, שוק נתוני העתק צפוי לגדול ב-20%
- עד 2020, כל אדם ייצור 1.7 מגה בייט בשנייה אחת
- משתמשי האינטרנט ייצרו כ-2.5 קוֹנְטֵי־לִיּוֹן (מיליארד מיליארדים) בייט של נתונים בכל יום
- עד 2020, יהיו כ-40 טריליון ג'יגה בייט של נתונים (40 zettabytes)
- 90% מכלל הנתונים נוצרו בשנתיים האחרונות
- בשנת 2012 היו לנו 2.5 מיליארד משתמשי אינטרנט, בשנת 2014 היו 3 מיליארד משתמשים ובשנת 2019 יש לנו 4.1 מיליארד משתמשים באינטרנט
- היום ייקח לאדם כ-181 מיליון שנים כדי להוריד את כל הנתונים מהאינטרנט (במהירות הורדה ממוצעת של 46Mbps)
- בשנה הבאה, כמות השימוש בנתוני האינטרנט לדקה יגיעו ל-3, 138, 420 GB

AI בישראל



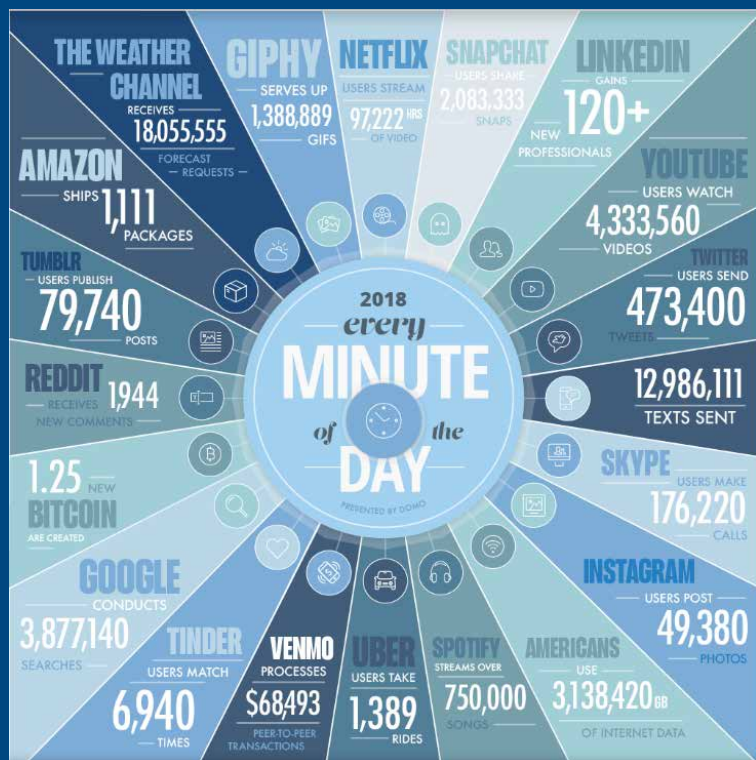
- בסוף שנת 2018 היו בישראל 6,673 חברות פעילות של טכנולוגיות עילית וחברות הזנק
- הבינה המלאכותית (AI) היא המגמה המובילה בקרב חברות הזנק, כאשר 17% מכלל חברות ההזנק בישראל פועלות בתת-מגזר זה
- בסוף שנת 2018 היו בישראל 1,150 חברות AI, לעומת 512 בלבד בשנת 2014
- חברות AI גייסו 2.25 מיליארד דולר ב-2018, לעומת 1.33 מיליארד דולר ב-2017 ורק 500 מיליון דולר ב-2014
- בשנה שעברה חברות AI היוו 37% מסך הגיוס של חברות ההזנק
- חברות הזנק וטכנולוגיות עילית ישראליות גייסו 6 מיליארד דולר ב-681 סיבובים ב-2018, עלייה של 15% מ-2017. רבעון ראשון \$ 1.55 מיליארד (+28% לתקופה המקבילה 2018)

מתוך:

<https://en.globes.co.il/en/article-start-up-national-central-6673-startups-in-israel-1001277453>

<https://finder.startupnationcentral.org>

כל דקה





ניצן גולדנברג

ניצן גולדנברג מזה 5 שנים בתחום בדיקות התוכנה, תפקיד נוכחי מהנדס בדיקות בחברת [SeatGeek](#), המוביל הראשי של קבוצת המיטאפ [TestIL](#), מרצה בקורסים לבודקי תוכנה וחבר בצוות המייעץ של [ITCB](#).



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 קבוצת QA ו-DevOps. חבר מייעץ למועצת מנהלים של [ISTQB](#) ומרצה בפקולטה להנדסת תוכנה במכללת [SCE](#).



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.



דורון בר

מכב משנת 1998 ברשימת המבוקשים של הבאגים. הוא לחם בחזיתות רבות (חברות הזנק וחברות גלובליות) בדרגות שונות (בודק, ר"צ, מנהל ועוד). ביום דורון מנהל מתודולוגיות בדיקה בחברת [Verint Systems](#) ובלילה [בלוגר](#).



ליז לירז גל

בעלת האתר [be-qa.com](#) ומפעילת מיזם "בודקים בקפה - חיפה" לחיזוק בודקים בתחילת דרכם.



עמית ורטהיימר

בודק תוכנה ב-[Deep Instinct](#).



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.



טל פאר

בעל ניסיון של מעל 20 שנים כבודק ומנהל בדיקות במגוון חברות וטכנולוגיות במודלי פיתוח שונים. טל עובד כיועץ עצמאי ומדריך בחברה [Practical Testing](#) שהקים, כשהוא מלווה ארגונים בהקמת צוותי בדיקות, הטמעת מתודולוגית בדיקות, שיפור תהליכי בדיקה והדרכות בדיקות. טל הינו חבר בצוות המנהל של [ITCB](#) וגזבר ארגון [ISTQB](#).



למה לכם לחשוב אם פספסתם!?!?

אם אתם רוצים
שהגיליון הבא
של מגזין עולם
הבדיקות
יגיע אליכם - לחצו
על הלינק להרשמה

bit.ly/TW-Reg

הרשמה