מגזין
רבעון שני 2021 גיליון מס' 25

עולם הבדיקות

www.testingworld.co.il

דבר העורך | ניצן גולדנברג

ניצן גולדנברג

מזה 6 שנים בתחום בדיקות התוכנה, תפקיד נוכחי מהנדס בדיקות בכיר בחברת SeatGeek, המוביל הראשי של קבוצת המיטאפ TestIL, מרצה בקורסים לבודקי תוכנה וחבר בצוות המייעץ AB של ITCB®.

קוראים יקרים,

"Only I can change my life,
No one can do it for me"
Carol Burnett

חשבתי רבות מה אוכל להעביר לכם הקוראים בגליון זה אשר תוכלו לקחת עמכם והחלטתי ללכת לתקופה של לפני תחילת הלימודים שלי, ליום שהחלטתי לשנות את חיי. לפני כ-6 שנים לקחתי החלטה שאחרי 20 שנה בעולם המלונאות הגיע הזמן לעשות שינוי בחיים ולהיכנס לעולם ההיי-טק. התלבטתי לאיזה תחום כדאי לי להיכנס, לפיתוח תוכנה?, אולי לעיצוב גרפי? אם כבר עיצוב גרפי אז למה לא עיצוב UI\UX? אבל לא חשבתי בכלל ללכת לכיוון של בדיקות תוכנה.

חמותי לחצה עלי ללכת ללמוד בדיקות תוכנה כי היא מכירה מישהו שלמד גם והוא מצליח מאוד בתחום. אמרתי לעצמי שאם הוא יכול אז גם אני יכול ולכן הלכתי בעיניים עצומות ללמוד בדיקות תוכנה.

ביום הראשון ללימודים התחברתי לכל הקבוצות פייסבוק לבודקי תוכנה בשביל ללמוד על המקצוע שאני מתחיל ללמוד ויודעים מה? התאהבתי!!!. מאותו הרגע, התחלתי ללמוד כל מה שיש על התחום הזה, השקעתי בלימודים מתוך התעניינות והצלחתי בלימודים. במהלך הלימודים התחלתי לעשות פרויקטים באתרי המונים בשביל לצבור ניסיון, התחלתי לעזור לחברי בכיתה בדברים שהם התקשו. מיד בתום הלימודים התחלתי את עבודתי הראשונה בתחום וכיום יש מאחורי רזומה מכובד ביותר בכל הקשור לעולם הבדיקות.

עד היום חמותי מספרת לכולם שאם לא היא, לא הייתי מגיע היום לאן שהגעתי, אבל אתם יודעים מה? אני לא הייתי מגיע לאן שהגעתי היום ולא הייתי יושב פה ומספר לכם את הדברים הללו אם לא הייתי משקיע בעצמי, אם לא הייתי יושב ומשקיע בלימודים, אם לא הייתי אוהב את המקצוע הזה "בדיקות תוכנה", אם לא היה לי חשוב לעזור לאחרים כי בסופו של דבר, רק אני יכול לשנות את חיי ואף אחד לא יכול לעשות זאת בשבילי.

מונח לפניכם גיליון מספר 25 של מגזין "עולם הבדיקות".
בגיליון זה תוכלו למצוא מגוון רחב של מאמרים חדשים, בנוסף לטורים המעולים והקבועים שלנו:

"מבדיקות Web לבדיקות CRM" מאת ליאור שם טוב

יאיר נסימוב מביא לנו עוד מאמר מהניסיון שלו והפעם "על הקשר בין חקלאות מודרנית לבדיקות מובייל"

מהנסיון שלי כמראיין אני מביא לכם מאמר על "טיפים להכנה לראיון מקוון"

כמו כן תוכלו להנות מהטורים הקבועים שלנו: ראיון עם מנהל/ת בדיקות, האנציקלופדיה לבדיקות, מחפש צרות, בחן את עצמך, סקירת כלים, הקופסא ומתודולוגיה ותהליכי בדיקות.

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

קריאה מהנה,
ניצן גולדנברג

עדכונים

- **כנס Testing & Automation Geek Week 2021 מתקיים גם השנה, לפרטים והרשמה יש להיכנס: קישור**
- **מוקדמות תחרות הבדיקות ISTC 2021 מתקיים בתאריך 23 באפריל 2021, לעדכונים הכנסו לפורום של TestIL בפייסבוק.**

תוכן העניינים

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, *Nokia*

עורך
ניצן גולדנברג, SeatGeek

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

יצירת קשר
אימייל:
info.testingworld@gmail.com

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

www.testingworld.co.il

www.itcb.org.il

כלי לבדיקות Testim | רחל ברוך

במאמר זה אני רוצה להכיר לכם את Testim, כלי לניהול בדיקות אוטומטיות תוצרת כחול לבן. כלי זה הוא כלי ניהול בדיקות אוטומציה מקצה לקצה, המבצע כמות גדולה של בדיקות ביציבות ובמהירות.

רחל ברוך

הנדסאית תוכנה, לפני 3 שנים לאחר הפסקה של שנים מעולם הה'ייטק חזרה לעולם התוכנה בכל הכוח. נהנית להיות בצד הבודק עם חשיבה של מפתחת. אין יום שהיא לא לומדת משהו חדש בעולם התוכנה.

הכרות testim

אם יתרונות של בדיקות אוטומציה הן מניעת טעויות אנוש וחסכון בזמן, אז Testim בא לקצר לנו את הזמן הזה בהרבה על ידי אלגוריתם למידת מכונה ובינה מלאכותית. הכלי יכול לסרוק כ-1000 בדיקות בשתי דקות.

Testim הוא כלי ניהול בדיקות לאוטומציה שמכיל את ניהול הבדיקות וביצוע ההקלטות ביחד. ניתן להשתמש בכלי ללא כתיבת קוד או ע"י כתיבת קוד ב-Java Script לצורך התאמה אישית. בדרך כלל כשרוצים לבצע בדיקות אוטומציה רגילות יש צורך בכלי נוסף עבור ניהול הבדיקות, ב-Testim כלי ניהול בדיקות נכלל.

הכלי מגיע כתוסף לדפדפן כרום וניתן להריץ את היישומים במערכות הפעלה שונות ובדפדפנים שונים. הכלי מגיע גם בגרסה חינמית וגם בתשלום וזה לפי הצורך האישי שלכם.

תכונות מרכזיות

1. כתיבה מהירה - Fast authoring
2. ניתוח שורש הבעיה - Root Cause Analysis
3. יציבות
4. גמישות
5. ביצוע
6. התממשקות
7. ניהול בדיקות
8. דוחות
9. התאמה לארגון - Enterprise readiness
10. תמיכה

כתיבה מהירה Fast Authoring

אחת התכונות הבולטות של הכלי היא היכולת לכתוב בדיקות בצורה מאוד מהירה. ניתן להתחיל להשתמש בכלי ללא כתיבת קוד, ניתן לבצע הקלטה ביישום הנבדק בדפדפן על ידי ביצוע הפעולות הנדרשות לצורך הבדיקה, לאחר מכן לסגור את הדפדפן ולשמור. ניתן להציג את כל השלבים שהוקלטו בנפרד ולבצע פעולות שונות כמו לשנות הגדרות, למחוק/להוסיף בדיקה, לצפות בצילום מסך, ליצור קבוצה, להוסיף ולידציה ועוד.

הכלי יודע לקחת את האלמנט שנמצא ביישום שנבדק לפי תכונות ב-DOM ונועל אותו על ידי לוקייטורים חכמים שתפקידם לנהל את מאות התכונות של אלמנטים ומדרג אותם כדי שהבדיקה לא תיפול במקרה של שינוי תכונות האלמנטים. דוגמא: אם מתכנת משנה את מיקום או צבע של האלמנט בקוד, ב-Testim הבדיקה לא תיפול כמו במקרה של בדיקות אוטומציה רגילות ולא יהיה צורך בשינוי הקוד. לוקייטרים חכמים הופכים את הבדיקות שלנו ליציבות מאוד, עם זאת הכלי מאפשר לנו לשנות את ההגדרות של תכונות האלמנטים.

לכלי יש מאפיינים מוכנים הניתנים לשימוש:

יצירת קבוצה: זהו מאפיין מאוד חשוב של הכלי. ניתן ליצור קבוצה לחלק מהשלבים (test steps) ולהשתמש בהם בהמשך. ניתן ליצור קבוצה בתוך קבוצה. יצירת קבוצה יכולה לחסוך שימוש חוזר של שלבים בבדיקה. לדוגמא אם נעשה שינוי בקבוצה אחת אז השינוי יתעדכן בכל הקבוצות מבדיקות אחרות.

ולידציה: ניתן להגדיר ולידציה עבור נתונים, דוא"ל (הכלי מייצר מיילים בתחילת הטסט ומוחק בסוף ההרצה), קבצי וורד, קבצי PDF ועוד.

תנאים: ניתן להגדיר תנאים לשלבים (steps) או לקבוצות.

לולאות: אפשרות הרצה מספר פעמים עד קבלת תוצאה הצפויה.

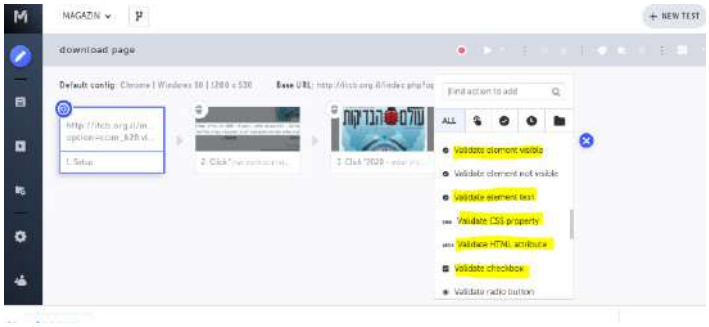
ניתן לעשות לולאה על רשימת אלמנטים.

פרמטרים: ניתן להגדיר פרמטרים ולקרוא להם מתוך קוד ג'אווה סקריפט או HTML.

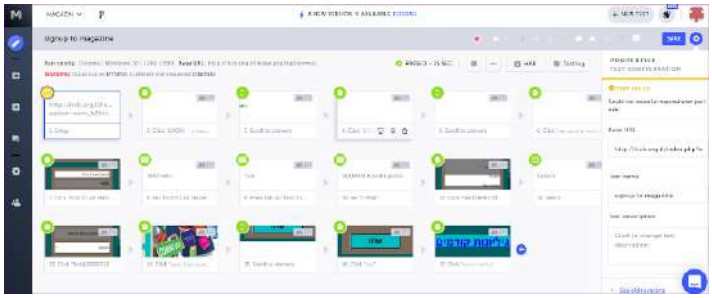
DDT: ניתן לשלב את נתוני הקלט מקבצי JSON, אקסל או CSV.

עם זאת ניתן לכתוב קוד לבדיקה בהתאמה אישית. הדרך הכי מהירה בכתיבת קוד היא קודם להקליט את שלבי הבדיקה ואז לייצא את הבדיקה לקוד. אפשרות נוספת לכתיבת קוד לבדיקה הוא להשתמש ב- Custom Steps שיכילו קוד JavaScript. קוד זה יורץ כחלק מהבדיקות ויאפשר גמישות מרבית במגוון הפעולות הנדרש.

מאפיינים של השלבים בבדיקה:



שינוי הגדרות של שלבים בבדיקה:



ניתוח שורש הבעיה – Root Cause Analysis

הכלי נותן יכולת למצוא במהירות את סיבה לכל נפילה על ידי צילומי מסך, קבצי network log-HAR ו-console logs. ניתן לשתף את תוצאות הבדיקות ולגשת לצילומי המסך וללוגים באופן מקוון.

יכולות אלו נמצאות גם במוצר נוסף בקוד פתוח שפיתחה חברת Testim ושנקרא Root Cause

Root Cause הוא פרויקט קוד פתוח שעוזר למצוא את שורש הבעיה במהירות בזמן הנפילה בבדיקה.

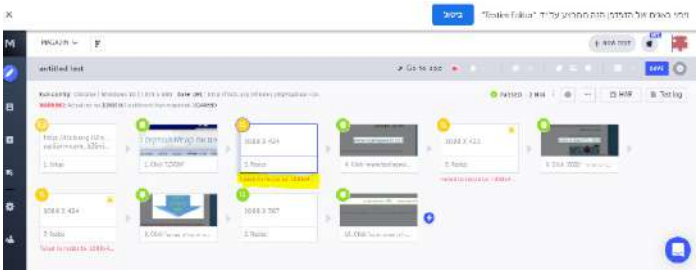
Root Cause יכול להשתלב עם טסט ראנר או פריימוורקים.

פריימוורקים כמו Puppeteer או Playwright הם מהירים ושימושיים, עם זאת בזמן הבדיקה, כאשר מתרחשת נפילה מציאת סיבת הנפילה לוקחת זמן.

כלי לבדיקות Testim | רחל ברוך

Root Cause מסייע למשתמשי Puppeteer ו Playwright למצוא את שורש הבעיה, ובכך לפתור את הבעיה במהירות. בזמן הרצת הבדיקה הוא לוקח צילומי מסך, קבצי network log-HAR ו-Console logs ושומר אותם במחשב או בענן. ניתן לשתף את תוצאות הבדיקות און ליין ולגשת לצילומי מסך וללוגים באופן מקוון. בנוסף גרסת הענן מאפשרת גישה לרשימות בדיקה, היסטורית ריצות ,לנתונים סטטיסטיים מצטברים ועוד.

מקרה נפילה :



צילומי מסך:

יציבות

אפשרות המתנה בהופעת אלמנט בדפדפן (סנכרון) מתבצעת בתנאים visible, non visible, time או התאמה אישי.

ניתן לשנות את הלוקייטרים באופן ידני.הבדיקות מתאימות את עצמן לשינויים ביישום וכך נמנעים כשלים באופן אוטומטי על ידי שימוש בבינה מלאכותית .

הצעת reusable component - הכלי סורק את הטסטים, מזהה קטעי קוד דומים, וימליץ להשתמש בקומפוננט קיים לפי צורך, או יאפשר לייצר ולעשות refactor בצורה אוטומטית. במידה ולא משתמשים ב-reusable component, תיקון שגיאה בבדיקות/עדכון מסוים צריך להעשות בכל מקום שבו נעשתה השגיאה, זה יכול להיות רלוונטי לבדיקה אחת מסוימת, אבל זה עשוי להיות רלוונטי לעשרות בדיקות נוספות, שבכל אחת מהן יש צורך לעדכן ולתקן את הדרוש תיקון. אם משתמשים ב-reusable component, יש לבצע את התיקון רק ב-group המסוים, ומייד התיקון חל על כל עשרות הבדיקות שמשתמשות בקבוצה זו. החברה מאפשרת ללקוחות משלמים לקבל הצעות ליצירת קבוצה על בסיס צעדים שחוזרים על עצמם בכמה טסטים. דבר זה מאפשר לעשות refactoring בצורה אוטומטית ויעילה. הלוקייטורים החכמים מגבירים את יציבות הטסטים בצורה רבה.

גם היכולת לייצר reusable component עוזרת לייצר בדיקות יציבות שקל לתחזק אותן. כמו כן - היכולת לאתר בעיה בבדיקה מסוימת בצורה מהירה-מקלה על התחזוקה של הטסטים ומגבירה את יציבותם.

התאמה ידנית של לוקייטורים:

גמישות

על ידי הוספת קוד ניתן להרחיב את הפונקציונליות של הבדיקה. ניתן להוסיף שלבים בהתאמה אישית ב-JS להרצת קוד בתוך או מחוץ לדפדפן. ניתן לייצר כתובות אימייל רבות לצורך ולידציית אימיילים בבדיקה. תהליך ההקלטה וטיפול בבדיקות כולל כשלים הינו קל להבנה. ניתן להקליט גם באמצע הבדיקה הקיימת. ניתן להציג כל שלב שהוקלט בנפרד ולבצע פעולות, כמו שינוי הגדרות, מחיקה, צפיה בצילום מסך ועוד.

ביצוע/הרצה

כאשר מריצים במקביל תוך התאמה של דפדפנים מותאמי grid אין האטה במהירות הביצוע.

ניתן להריץ בדפדפן כרום או בדפדפנים נתמכים ב-Testim grid, Sauce Labs, Browserstack, 3rd party grid או ב- Selenium-based grid.

הכלי מציע שימוש ב - mock network traffic של AUT כחלק מהבדיקה. במשך הבדיקה במקום לבצע תקשורת רגילה המערכת מעכבת את התקשורת ותחזיר mocked response.

דפדפנים נתמכים ב-Grid:

Browser	Testim Grid	Selenium Grid	3rd Party Grid
Chrome	Yes - Linux	Yes - multiple OS	Yes - multiple OS
Firefox	Yes - Linux	Yes - multiple OS	Yes - multiple OS
Safari	Yes - macOS	Yes - macOS	Yes - macOS
Edge	Yes - Windows	Yes - Windows	Yes - Windows
Edge Chromium	Yes - Windows	Yes - Windows	Yes - Windows
IE 11	Yes - Windows	Yes - Windows	Yes - Windows

התממשקות

ניתן לממש התממשקות עם כלי דיווח מעקב באגים מפורסמים כמו Jira, Trello, Slack, Github Code Management.

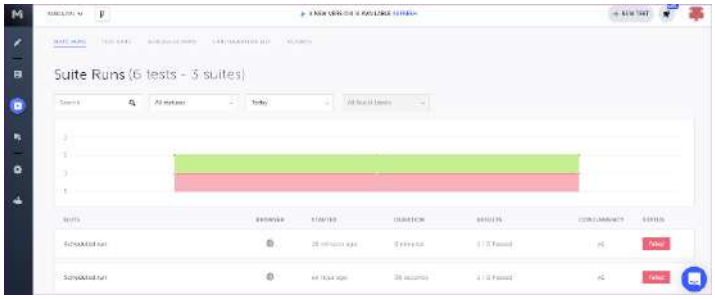
יווצרו בהם עם פרטי תיאור וצילומי מסך של הבאגים.

התממשקויות נוספות:

- ניהול קוד - API של הכלי לניהול בראנצים, Github Bitbucket
- ולידציית פיקסל, Applitools
- ניהול בדיקות התממשקות, TestRail

```
Shell
npm install -g @testim/testim-cli
```

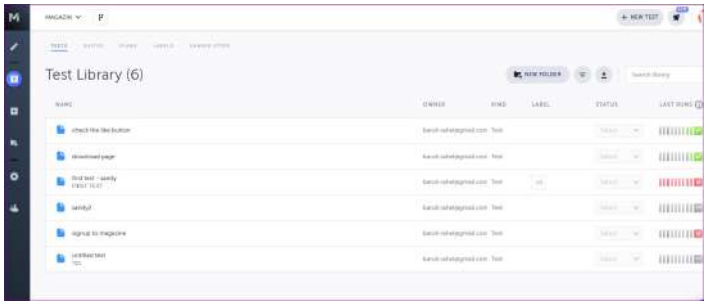
התממשקות של הבדיקות ל-CI מתבצע על ידי Testim CLI (ב-npm). מעקב לאחר תרחיש הבדיקה מראה מה עבר/נכשל ופותח אותו בלחיצת כפתור ב-test result. תוצאות ההתממשקות יועברו בסטנדרט JUnitXmlReport ויופיעו בדשבורד. התממשקות מ-CLI עם:, Azure Devops build pipeline, Bamboo, Circle CI, Codeship Jenkins עם Docker, TeamCity, VSTS ו-GitLab, TFS.



כלי לבדיקות Testim | רחל ברוך

ניהול בדיקות

אם רוצים להריץ את הבדיקות בסדר מסוים ולא במקביל, ניתן להשתמש ב-test suite או labels. כמו כן ניתן להוסיף מספר רב של slebal לכל בדיקה ברשימת הבדיקות.



לעבודה עם לייבלים יש שתי מטרות: לארגן את הבדיקות ולשלוף אותם לפי הצורך על ידי סינון וגם על ידי יצירת test suites, שונים להריץ מקרי בדיקות ב-CI. לדוגמא, לייבל בשם sanity ירוץ לאחר כל שינוי בקוד, ובשם monitor ירוץ כל 15 דקות כדי לוודא שיישום בפרודקשיין עובד תקין. ניתן להריץ ב- CLI את הלייבל שרוצים בצורה:

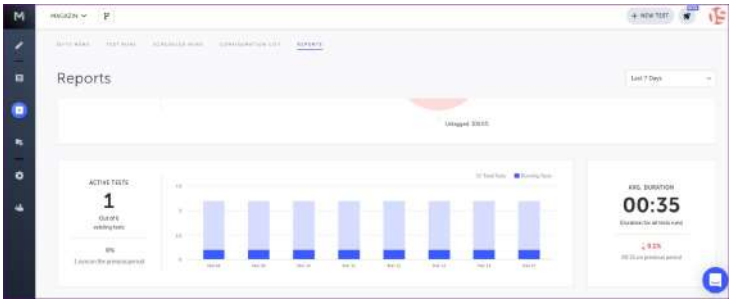
```
testim-- label>" YOUR LABEL-- "<token>" YOUR ACCESS TOKEN-- "<project>" YOUR PROJECT ID-- "<grid>" Your grid name-- "<report-file test-results/testim-tests-report.xml
```

בקרוב:

- **מעגל חיים ניהול אוטומטי** - פתרון שלם של ניהול בדיקות אוטומטי. ניהול מאות ואלפי בדיקות יהיה יותר קל כאשר מבינים איזה בדיקות לא יציבות, אילו בדיקות שבורות, וכמובן מי אחראי לתקן.
- קבלת אישור לפני merge של בראנץ (Pull Request) כדי להגן על בראנץ הראשי.

דוחות

דוחות מציגים מידע על מספר הרצות, כמות הבדיקות שעברו/נכשלו, איזה בדיקות פעילות כרגע, הצגת מידע על מצב flaky שדורש התייחסות. ניתן לצפות בהיסטורית ההרצה לפי סיבת הכשל. ניתן לסנן דוחות לפי label/tag/other. בנוסף מתקבל דיווח במייל מדי שבועי.



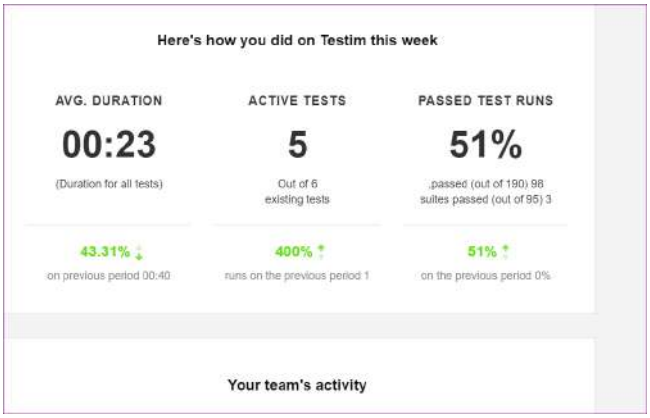
התאמה לארגון
Enterprise readiness

שירותים למוצר בהתאמה אישית לארגון:

- הרשאות ברמת הפרויקט או הארגון
- זימון משתמשים לפרויקט קיים
- גישה לבראנץ ללא אפשרות שינוי בו
- SSO
- API עבור בראנץ
- פרוטוקולים לנגישות כמו WCAG, ARIA ועוד
- VPN, SOC2 type II

תמיכה

בכל אורך הזמן שעבדתי עם הכלי קיבלתי מענה מיידי לשאלות ששאלתי ובנוסף קיבלתי מיילים מהחברה על מצב הבדיקות שביצעתי עם הצעות לשיפור הבדיקה. תמיכה טכנית נותן מענה 24/7. בנוסף התיעוד באתר החברה מכיל מידע מספק.



יתרונות

- קל לתחזק עקב השימוש בבינה מלאכותית
- הצגת כישלונות במסך ההקלטה על ידי חקר וחיפוש בעיית השורש root cause analysis-
- ממשק משתמש אינטואיטיבי
- ניתן לבצע מספר רב של בדיקות בזמן קצר
- אפשרות ליצור קבוצות לצורך שימוש חוזר של השלבים בבדיקה נוספת
- הרצה מדפדפן או מ-CLI
- התממשקות עם מערכות מעקב באגים פופולריות
- התאמה אישית להרחבת בדיקה
- אפשרות יצירת מספר רב של מיילים לצורך ולידציה
- הכלי מאפשר להריץ במקביל מספר רב של התקנים בדפדפנים שונים
- תמיכה טכנית 24/7 ומענה מהיר

חסרונות

- לא ניתן לשנות בעלים לבדיקה
- ניתן להוסיף Custom Steps שכתובים ב-JS ולא בשפה אחרת

ציון:

מענה לצרכים של החברה 9.9/10

נותן למשתמש חווית שימוש 9.8/10

תמיכה וקהילה 10/10

סה"כ 9.9

בחן את עצמך | טל פאר

אחת השאלות הראשונות שסטודנטים שואלים אותי בקורס שמכין לבחינת ISTQB® היא "מה כוללת הבחינה?". האם כל תוכן הבחינה נמצא בסילבוס?

הסילבוסים של ISTQB® מכילים לא מעט ידע שאותו צריך לדעת כדי לעבור את הבחינה, אבל לא תמיד ברמה מספיק מפורטת. את הידע הדרוש מקבלים בקורסים מתאימים או בקריאה של ספרות מקצועית.

איך בכל זאת נדע מה תכלול הבחינה? בתחילת כל פרק מופיעה רשימת יעדי לימוד (learning objectives) המתאימים לכל פרק. לכל יעד לימוד מוגדרת רמת ידע קוגניטיבית (K-level) שמגדירה את הנדרש בשאלה. הסברתי בפירוט על רמות הידע הקוגניטיביות בגיליון 2. כל שאלה בבחינה צריכה להיות מקושרת ליעד לימוד מוגדר ולהתאים לרמת הידע המוגדרת. החל מגיליון זה, בפתרון השאלה אסביר לאיזה יעד לימוד ורמת ידע מתאימה השאלה.

הסילבוס מגדיר תהליך בדיקות (test process) שמורכב ממספר רבות בדיקה (test levels). השאלה שלנו היום מדברת על ההבדלים בין רמות הבדיקה השונות.

המשפטים הבאים משווים בין בדיקות רכיבים (component testing) ובדיקות מערכת (system testing). איזה משפט ממשפטים אלה נכון?

א. בדיקות רכיבים מוודאות את הפונקציונליות של יחידות תוכנה שניתן לבדוק בנפרד; בדיקות מערכת מוודאות את הממשקים בין רכיבים ובין חלקים שונים של המערכת.

ב. מקרי בדיקה (test cases) לבדיקות רכיבים נגזרים בדרך כלל ממפרטי הרכיב, מפרטי העיצוב או מודלים של הנתונים; מקרי בדיקה לבדיקות מערכת נגזרים בדרך כלל ממפרטי דרישות או מקרי שימוש (use cases).

ג. בדיקות רכיבים מתמקדות רק על מאפיינים פונקציונליים; בדיקות מערכת מתמקדות במאפיינים פונקציונליים ומאפיינים לא פונקציונליים.

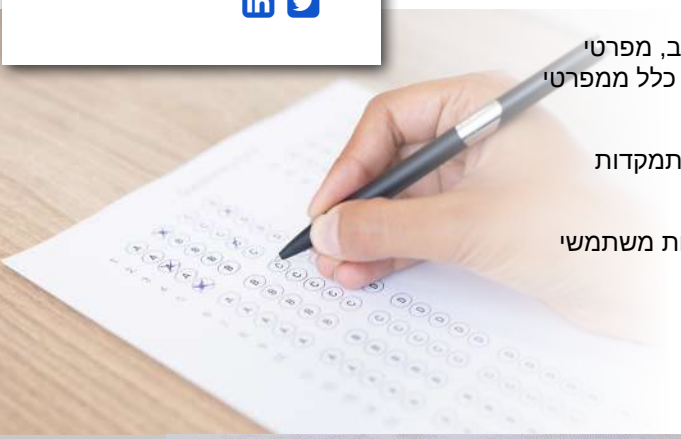
ד. בדיקות רכיבים הן באחריות הבודקים; בדיקות מערכת הן בדרך כלל באחריות משתמשי המערכת.

*לצפייה בתשובה המפורטת - דפדפו לעמוד 17

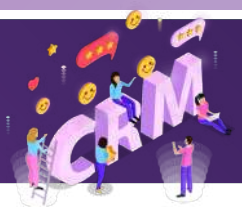
טל פאר

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

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



מבדיקות Web לבדיקות CRM | ליאור שם טוב



במאמר זה אסקור את **חוויית המעבר של בודקים בעלי רקע עיקרי בבדיקות מערכות WEB עם פיתוח IN HOUSE, לבדיקת מערכת CRM (מוצר מדף מספק חיצוני).** בודקים החווים לראשונה כניסתה של מערכת מרכזית לארגון שהיא מוצר מדף, עוברים תהליך של שינוי, חידוש, למידה ואתגרים. הכתבה תתמקד בבודקים אשר היו מורגלים במשך שנים רבות לעבוד עם מערכות WEB בפיתוח פנימי, ללא היכרות עם מערכת CRM או מערכות דומות ולראשונה מקבלים מוצר מדף הדורש שכלול בתפיסת הבדיקות. אסייג במעט את המונח "חוסר היכרות" ואציין שבמערכות עם פיתוח פנימי לעיתים יש עבודה עם **מוצרי תשתית** שגם הם מוצר מדף, למשל עבודה עם תשתית kendo או מנוע חיפוש/חוקה וכיו"ב שהם מנת חלקם של הרבה בודקים אשר התנסו בבדיקות מוצרים אלו או מוצרים דומים. למעשה ב"מוצרי תשתית" מדובר בהתמודדות זהה בדומה לכל מוצר מדף, אבל ברמה מינורית ביחס לכלל המערכת, ההתמודדות ממוקדת למוצר עצמו שמוכל בתוך המערכת עצמה ומהווה את אחד הרכיבים בה (ולא מצב בו כל המערכת היא מוצר מדף), ולכן בתפיסת הבדיקות, התייחסות לכלל המערכת לא משתנה, והשינוי, אם קיים כזה, הוא רק ביחס למוצר המדף התשתיתי, ולכן הבודק לא חש בשינוי משמעותי מבחינת התמונה הכוללת של המערכת.

במאמר לא אתייחס להיבטים של יתרונות וחסרונות או בעד ונגד CRM, אלא אתמקד בהתמודדות הבודקים הניגשים לבדוק לראשונה מערכת CRM.

מבוא

מערכת CRM הינה פלטפורמה לניהול מערכות ארגוניות המפשטת את התהליך העסקי וממשת אותו עם אובייקטים עסקיים, כאשר מעליהם בנויים קשרים, תהליכים, דוחות ועוד. ה-CRM מאפשר הקמה פשוטה של מערכת על התשתית שהוא מספק, קרי מתן כלים ויכולות נרחבות, אינטגרציה מול מערכות קיימות בפיתוח פנימי, התממשקות למוצרי מדף שונים, עדכוני גרסא מהירים ועוד.

ארגונים אשר מכניסים מערכת CRM או מוצר מדף אחר לצד מערכות בפיתוח עצמי, מהווים לגוף ה-IT אתגר גדול. ה-IT בארגון נדרש לבצע את ההתאמות לחלק או לכל המערכות, שירותים, ממשקים, מסדי נתונים, התממשקות וכיו"ב. לעיתים מדובר בהכנסת מערכת CRM חדשה לצורך עסקי חדש, ולעיתים בהחלפת מערכת קיימת. למעשה בשני המצבים מדובר בשינוי רציני ורגשי, הדורש תהליכים ארגוניים מורכבים למשל: מנהלי הארגון נעזרים במומחים מתחום הייעוץ הארגוני אשר מובילים תהליך של שינוי והטמעה למשתמשי הארגון כדי לייצר נחיתה רכה למשתמשים ביום בו תגיע המערכת לשימושם; ה-IT עובר גם הוא שינוי בכל הרמות, בהיבט התשתיתי כפי שתואר לעיל וכפי שיתואר בהמשך, וברמות השונות בארגון למשל: הנהלה, מנתחי מערכות, צוותי פיתוח ובדיקות אשר מתאימים עצמם לעבודה עם מוצר מדף, גיוס אנשים עם ניסיון ב-CRM ו/או הכשרת צוותים קיימים, ביצוע הדרכות ע"י הספק, עמידה בדרישות אבט"מ, הסבות נתונים והיבטים נוספים אשר יריעת כתבה זו קצרה מלהכיל.

בודקים החווים לראשונה כניסתה של מערכת מרכזית לארגון שהיא מוצר מדף, עוברים תהליך של שינוי, חידוש, למידה ואתגרים

במהלך שנות עבודתי פגשתי מערכות CRM בכמה צמתים, בכל צומת הבודקים היו עם ניסיון בבדיקות מערכות CRM, או כהיכרות יחידה של מערכות מסוג זה או כהיכרות נוספת לצד המערכות בפיתוח פנימי הקיימות בארגון. הכתבה תתמקד בצומת בו ניהלתי צוות בודקים אשר נפגשו לראשונה עם מערכת CRM ושם חוותי לראשונה תהליך של הובלת שינוי תודעתי מתפיסת בדיקות מוכרת וידועה לתפיסת בדיקות משוכללת.

במאמר אתאר את התהליך העובר על הבודק במפגש עם מערכת CRM במהלך פרוייקט בדיקות במקטעים הבאים:

1. **אפיון וכתיבת תסריטים**: צוות הבודקים נפגש לראשונה עם אפיון של מערכת CRM באמצעות קריאה מודרכת, במטרה לזקק את הדרישות הרלוונטיות לכתיבת התסריטים.
2. **בדיקות**: התמודדות עם מערכת שהיא מוצר מדף.
3. **תקלות**: זיהוי תקלות אשר מהוות ערך לצוותי הפיתוח וסינון תקלות לא רלוונטיות או שתיקונם מתקיים במימד חיצוני לארגון.
4. **אוטומציה**: מיכון מוצר מדף ושכלול דרכי הזיהוי של אלמנטים.
5. **סיכום**: הצגת מסקנות והמלצות לצוותי בדיקות.

אפיון וכתיבת התסריטים

במערכות עם פיתוח פנימי הבודק מתמודד עם אפיונים אשר מתארים את דרישות המערכת מקצה לקצה, לעומת זאת העבודה עם אפיון של מערכת CRM נראה קצת אחרת מבחינת שמות ומושגים. האפיון מתמקד בעיקר בצד אשר יפותח מאשר בצד המובנה, פחות התייחסות ל-UI ויותר מיקוד בקונפיגורציה, לכן התמודדות עם ה"**חוסרים**" לכאורה, נאלץ הבודק להשלים באמצעות התסריטים (בודקים המנוסים בבדיקות CRM לעיתים יוותרו על השלמת חוסרים בתסריטים, מתוך הנחה שמי שמריץ את התסריטים כבר קיימת לו היכרות עם CRM), כלומר על הבודק להשלים מידע אשר לא קיים באפיון (לא בכדי בדרישות הגיוס מבקשים היכרות מוקדמת עם CRM). אפיון המערכת גורם לבודק לחוסר לוגיקה או הגיון מסויים בהקשר של בניית התסריט ולכן מוטב שבתחילת הדרך הבודק יקבל הדרכה ולמידה בה הוא יקבל את הידע והכלים להבין ולהכיר את תכונות ומאפייני ה-CRM ובהתאם לזה הוא יגשר על הפערים והחוסרים באפיון. בשלב של הקריאה המודרכת (לאחר שהבודק כבר קרא את האפיון ורשם הערותיו) הבודק מעלה שאלות רבות למאפיין או למיישם וכך מדרישה לדרישה צוות הבודקים בונה את אסטרטגיית הבדיקות, מיפוי הדרישות העסקיות, מבסס את עץ הבדיקות ומקרי הבדיקה עד לשלב כתיבת תסריטי הבדיקות.

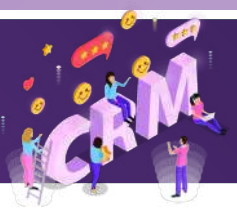
היכרות מקדימה עם מערכת CRM יש לה חשיבות באיכות התסריטים, ולכן מאוד מומלץ לפני כתיבת התסריטים, לקבל הדרכה מסודרת ע"י הספק או אנשים בארגון עם ניסיון בכלי, אשר יציגו לצוותי הפיתוח ובדיקות את התכונות הקיימות במערכת, גמישות לעומת קשיחות, מובנה לעומת חידוש, best practices ותרגול בכלי, בנוסף, לקיים עם מנתח המערכות קריאה מודרכת של האפיון. שילוב של הדרכה מצד ספק על המערכת וקריאה מודרכת של האפיון סוגר את הקצוות הנדרשות לבודק לצורך הבנת אסטרטגיית הבדיקות וכתיבת מיטבית של תסריטי בדיקה.

היכרות מקדימה עם מערכת CRM יש לה חשיבות באיכות התסריטים, ולכן מאוד מומלץ לפני כתיבת התסריטים, לקבל הדרכה מסודרת ע"י הספק או אנשים בארגון עם ניסיון בכלי

ליאור שם טוב

ניהול פרויקטים וייעוץ בתחום הבטחת איכות מערכות מידע מעל ל-20 שנה..

מבדיקות Web לבדיקות CRM | ליאור שם טוב



בדיקות

הגענו לשלב המסקרן בפרויקט בו הבודק לראשונה מתחיל להרגיש את מערכת ה-CRM. במהלך הבדיקות הבודק פוגש מעין עולם חדש (יקום מקביל) של מערכת שלא נראית כפי שהורגל עד היום בבדיקות מערכות בפיתוח פנימי, לדוגמא ישויות, חוקים עסקיים, עצמים, UI, אלמנטים, וגם תכונות של המערכת אשר פועלות באופן שונה או מצופה מהיקום המוכר, למשל "כפתורים", במערכת CRM נמצאים כפתורי פעולה, אשר לא תמיד יש להם שימוש, אבל הם שם, אם להסירם או לא זאת שאלה שמופנית למנתח המערכות או למיישם אשר לא תמיד ממהר להסירם (שינויים מובנים נעשים באמצעות שינוי הקונפיגורציה של המסך), סיבה שכיחה לאי הסרת ה"כפתורים" המיותרים, כי אין להם השפעה על התהליך העסקי (למשתמש הסביר זה יראה מוזר ומיותר שיש כפתורים במערכת שאין להם שימוש) החשש, שעם התקדמות פיתוח, המערכת תהיה בשלבים מתקדמים וכאשר תתקבל דרישה של הלקוח להסיר כפתורים מיותרים, אזי יהיה מדובר בתהליך ארוך ויקר בו יש לסרוק את כל המסכים ולבצע בחינה מחודשת איזה כפתור מיותר ואיזה לא (כאשר החלופה המיטבית לטפל בצורך עבור כפתורים מיד עם הקמת מסך חדש), לעומת זאת במערכת WEB המאפיין לא יאפיין כפתורי פעולה אם לא קיים להם צורך עסקי, יתרה מזו, מנתח המערכות גם לא יפתח עתודות של כפתורים לצורך שימוש עסקי אם אין להם צורך מיידי או ערך ללקוח. מצבים אילו חוזרים על עצמם גם ברמת מבנה UI, פקדים, פעולות עסקיות ועוד.

הבודק העובר בין המסכים מוצא עצמו משקיע פחות בבדיקות GUI (תסריטי GUI כמעט ולא נכתבים) או בפונקציונאליות מובנית, אלא בעיקר בחוקים העסקיים המגדירים את ההתנהגות של הישויות במערכת, ולידציות, בדיקת נתונים ומאפיינים המגדירים את התצורה הכללית של אותה ישות (לדוגמא – לאפשר בתוך ישות מסוימת לבצע שליחת מייל חיצוני), לעומת זאת במערכות עם פיתוח פנימי, כל מה שמפותח נבדק.

מבחינת UI במערכת CRM בעיקר במסכי הישויות (טפסים) או תצוגות (מסכים ראשים של הישויות המציגים את כל הרשומות של הישות וסטאטוס הישות) מוצאים שהם זהים או כמעט זהים אחד לשני ולמעשה מדובר בתחושות מאוד מונוטוניות אשר משעממות את הבודק. (באחת ההרצאות שנוכחתי על מערכת CRM, הספק ציין שאין לפתח ציפיות ל"יופי" במערכת או "נגישות" מבחינת UI, אבל תצפו לקבל מערכת מאוד יציבה).

התמודדות הבודק עם המערכת מאתגרת מאוד, לכן מאוד מומלץ, טרם הגעת המערכת לסביבת הבדיקות, שהבודק יבקר אצל צוותי הפיתוח ויבקש לראות את המערכת, וכך הוא נחשף לראשונה ל-UI של הכלי, חש לראשונה את המסכים, תצוגות, ישויות, חוקים העסקיים, מסכי ניהול ואת המודולים המרכיבים את המערכת.

תקלות

עם תחילת הבדיקות, הבודק מתחיל ללמוד ולהכיר את תפיסת המערכת, מבנה המערכת, תצוגות, מסכים ברמת המשתמש, מסכים ברמת מנהל מערכת, תפריטים, מיקום של ישויות, היבטים של פעולות עסקיות, קונפיגורציה, מודולים ומתחיל לזהות (גם תוך כדי ניסוי וטעיה) תקלות במערכת.

התקלות הנפתחות מחולקות ל-4 קטגוריות עיקריות:

א. תקלות רגילות קלאסיות (חוק העסקי, תצוגה או ולידציה בניגוד לאפיון או נתונים לא תקינים).

ב. תקלות בתצורת המערכת (תקלות קונפיגורציה)

ג. תקלות UI, אשר מטרתן בעיקר לבטא שיפור נדרש בנגישות המידע למשתמש (תקלות אילו מתגלות כאתגר פיתוחי ראשון במעלה, מהסיבה שמערכת CRM מאוד מגבילה את המתכנת בשינויי (UI).

ד. תקלות CRM (תקלות שמפנים לחברה שפיתחה את ה-CRM ואולי בגרסה מסוימת, בעתיד מסוים ינתן פתרון לתקלה ביחד עם כל משתמשי CRM בעולם).

כאשר פותחים תקלות מומלץ לסווג ו/או לתייג אותם בהתאם לקטגוריות

כאשר פותחים תקלות מומלץ לסווג ו/או לתייג אותם בהתאם לקטגוריות. חשיבות לכך גבוהה מהסיבה שבתקלות מתויגות כתקלות CRM או אלמנטים במערכת שהם מובנים ולא מוגדרים כתקלה "קלאסית" מאוד רצוי לשתף את יתר הבודקים; באמצעות שיתוף הידע, הבודקים ולומדים לזהות במהלך הבדיקות מתי מדובר בתקלה אמיתית ומתי לא וחוסכים זמן לצוותי הפיתוח בפתירת תקלות מיותרות, כלומר תקלות אשר הפיתוח לא יכול לתקנם בגלל האופן שבו מובנה ה-CRM.

אוטומציה

האתגר הראשון של צוות האוטומציה בהגיעו לְמַכֵּן פעולות עסקיות במערכת CRM, הוא זיהוי האלמנטים. בניגוד למערכות בפיתוח פנימי שניתן לגשת בקלות דרך קוד לאלמנטים במסך (המתכנת נותן לכל אלמנט שם מזהה שבאמצעותו האוטומציה יכולה לגשת לאלמנט), במערכת שהיא מוצר מדף הגישה לחלק מהאלמנטים היא רק דרך מעטפת ייעודית שאותו מוצר מספק וישנם מצבים בהם לאלמנטים אחרים אין גישה כלל. בסופו של תהליך ובעזרת התייעצות עם צוותי הפיתוח ופורומים רלוונטיים כל מכשול הוא שאיפה וכל שאיפה מקבלת ביטוי כפתרון וסיפוק אדיר לצוות האוטומציה .

לסיכום

לאחר סבב בדיקות ראשון, צוות הבודקים הרגיש כיצד מיומנות הבדיקות עלתה בכמה מונים. לאחר קבלת תכולת העבודה הנוספת הבודקים הרגישו כמו שחיינים מנוסים אשר שולטים בעוד סגנונות שחייה, המחכים בסקרנות לאתגר הבא במערכת, ואכן ישנם אתגרים נוספים (למשל: הסבות נתונים, שילוב בין הרשאות של ה-CRM למערכות האחרות בארגון, אינטגרציה עם מערכות אחרות, ממשקים פנימה והחוצה, עבודה עם SERVICES ועוד).

אחד הדברים המרשימים אשר התפתח בין צוותי העבודה היא שפה משותפת בין כולם אשר נסקה והתפתחה עם התקדמות פיתוח המערכת והכי חשוב הנאה לאורך כל תהליכי העבודה.

ראיון עם מנהלת בדיקות נטליה רחמני, ThetaRay

נטליה רחמני

שמי נטליה רחמני ואני בתחום הבדיקות קרוב ל 17 שנה. הגעתי לתחום ממש במקרה. אחת החברות שלי שעבדה בתחום זיהתה את היכולות שלי והמליצה לי על קורס בג'ון ברייס, כבר במהלך הקורס התחלתי לעבוד באלביט.

רוב הקריירה שלי מתמקדת בתחום הביטחוני, הרפואי והפיננסי. במהלך הקריירה שלי עבדתי באלביט בפרוייקט צ"יד (צבא יבשה דיגיטלי), ניהלתי בדיקות של כיפת ברזל במערכות של שליטה ובקרה בחברת אמפרסט (חברת בת של רפאל) במשך קרוב לשבע שנים ועוד שנתיים נוספות בחברות יותר קטנות אשר מתמקדות בתחום הבטחוני.

בנוסף לזה, עבדתי גם בתחום הרפואי בחברת Algotec. שם ניהלתי תחום בדיקות של מערכות מידע רדיאולוגיות - מערכות המתחברות ל-MRI, CT, אולטראסאונד ועוד.

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.

ספרי לנו מעט על התפקיד הנוכחי

בשנה וחצי האחרונות אני עובדת בחברת ThetaRay . ThetaRay היא הספקית המובילה של ניתוח ביג דאטה מבוסס בינה מלאכותית (AI). אנו מחויבים לסייע לארגונים פיננסיים להילחם בפשעי סייבר פיננסיים כמו הלבנת הון, הונאה והתקפות כספומט, המשמשות למימון טרור, סמים וסחר בבני אדם, עבדות מין ומעשים זדוניים אחרים. פלטפורמת הניתוח המתקדמת של ThetaRay מאפשרת ללקוחות לנצל את העוצמה והתחכום של האלגוריתמים שלנו לבניית יישומים מותאמים אישית - כדי לזהות חריגות במערכי נתונים גדולים, לא משנה מה המקור. הוא תומך במשתמשים מגוונים, החל ממדעני נתונים וכלה באנליסטים עסקיים, ומשתלב בצורה חלקה בתהליכי עבודה ומערכות קיימים.

בשנה וחצי האחרונות אני מנהלת קבוצת בדיקות בארץ וקבוצת בדיקות ב-offshore.

במהלך הקריירה ניהלתי קבוצות שונות ובגדלים שונים, הניהול היה ניהול מוצר וגם ניהול פרוייקטים.

איך בנויה קבוצת הבדיקות?

קבוצה הבדיקות הנוכחית מכילה קבוצת בדיקות אוטומציה בארץ וקבוצת בדיקות ב offshore. כל אחד מהבודקים הוא feature owner - כלומר, כל אחד אחראי על פיצ'ר מסוים מתחילתו, שלב של כתיבת מסמכי תכנון הבדיקות, בדיקות ידניות לפני כתיבת אוטומציה, כתיבת בדיקות אוטומטיות ועד סופו – שלב של מסירת הפיצ'ר.

מה הם האתגרים שלכם ואיך את צופה שזה ישתנה בעתיד?

כשנכנסתי לתפקיד, התחלפו לא מעט אנשים מכל מיני סיבות, הגיעו אנשים חדשים, הוקם צוות בחו"ל ולכן היו למעשה שני אתגרים מרכזיים.

בתחילת העבודה היה אתגר גדול בהכנסת מתודולוגית עבודה ובדיקות חדשות, סנכרון ופרוצדורות חדשות, כל זה הוכיח את עצמו כעוזר לצוות וכתוצאה מכך איכות הבדיקות עלתה באופן משמעותי.

אתגר נוסף היה הטמעת מתודולוגיית עבודה. כתיבת מסמכים כדי לראות שאנחנו לא מפספסים שום דבר, העלאת כיסוי הקוד והבדיקות. מנהל הפיתוח ואני הגדרנו שעל האוטומציה שרצה להיות יציבה כמה שאפשר וגרסה לא תשתחרר כל עוד ישנם טסטים שנופלים בסביבת האוטומציה ואנחנו עובדים קשה על ייצובם.

מבחינת כיסוי הקוד ישנו אתגר לא פשוט. הטסטים האוטומטים לא יכולים להגיע לכל שורת קוד ואנחנו צריכים להבין לאן אנחנו לא יכולים להגיע מבחינת אוטומציה ולבחון כיצד בכל זאת נוכל להגיע ולכסות בצורה אחרת גם את האזורים האלה.

בתחילת העבודה היה אתגר גדול בהכנסת מתודולוגית עבודה ובדיקות חדשות, סנכרון ופרוצדורות חדשות ,כל זה הוכיח את עצמו כעוזר לצוות וכתוצאה מכך איכות הבדיקות עלתה באופן משמעותי.

הלקוחות שלנו דורשים תהליכים מאוד מסודרים וסטנדרט מאוד גבוה כך שבזמן שחרור מוצר עלינו לדווח את אחוזי הכיסוי שלנו מבחינת הקוד ומבחינת הבדיקות, מה הרצנו ומה היו תוצאות ההרצה.

אתגר נוסף הוא ניהול צוות ב-Offshore. אומנם רובנו דוברי רוסית ואף חלק מהישיבות שלנו מתנהלות בשפה הרוסית אך עדיין ישנם קשיי טכנולוגיה, מרחק, מנטליות שונה ואתגרי סנכרון שהרי חצי מוצר מפותח ב-offshore וחצי מוצר מפותח בארץ ועליהם לתקשר יחד ולוודא ששום דבר לא מתפספס.

מה הם האתגרים הייחודיים לקבוצת הבדיקות שלכם בהתאם לתחום הטכני ולתהליכי הפיתוח שלכם?

אנחנו מוציאים גרסאות גדולות פעם בכמה חודשים ובכל גרסה יש שינוי טכנולוגי מאוד גדול מבחינת המוצר ולרוב גם מבחינת כלים על מנת להיות תמיד בקדמת הטכנולוגיה ולהכיר כלים חדשים. מעבר בין כלים דורש מאיתנו לשכתב את הקוד ולשנות תהליכים מבחינת האוטומציה כך שיתאימו למוצר החדש.

ע"י מעבר לכלים חדשים אנחנו נשארים אקטואליים מבחינת השוק, קל לנו יותר לגייס אנשים חדשים וזוהי גם דרישה שמגיעה מבחינת הלקוחות שלנו כדי שנשאר ברמת גבוהה. מעברים שכאלה דורשים מהצוות לגלות גמישות ורצון.

איך את מניעה את העובדים שלך ומה עוד היית רוצה לעשות?

ישנן הרבה מאוד דרכים להנעת העובדים. ישנו הנושא הכספי, יציאה לקורסים חיצונים, התפתחות אישית ועוד. מבחינתי, תחילה אני משתמשת במה שיש ברשותי ברמה הניהולית שלי ושואפת לפעול בגבולות היכולת שלי.

ראיון עם מנהלת בדיקות נטליה רחמני, ThetaRay

בסביבה אינטנסיבית של הפיתוח כל אחד מאיתנו נדרש לעמוד בדד ליין של שיחרור הגירסה או סיום כתיבת אוטומציה על פיצ'ר מסויים לכן צריך לדעת לנהל את זה נכון.

כאשר הגעתי לתפקיד התחלתי לשתף אנשים בתהליך קבלת החלטות, לשתף אותם בתהלכי עבודה ואחריות שלהם בתהליך, כל זה בהחלט הניע אותם בצורה נכונה ויפה וגרם להם להרגיש שותפים לחברה וללקוחות ובכך מוטיבציה שלהם עלתה באופן משמעותי.

היום הצוות שלי מגובש, מכיר צוותים אחרים בחברה ומשתף איתם פעולה. אנשים לא עובדים בשביל לקבל תלוש בסוף החודש אלא ממש מתוך מעורבות ורצון להצלחה של החברה.

דבר נוסף שחשוב לי הוא העלאת היכולות הטכניות של הצוות. מעבר מתמיד בין מוצרים דורש מאיתנו להסתגל לטכנולוגיות חדשות יחסית מהר וצריך לדעת איך לעשות את זה. לא בהכרח צריך להוציא מיד את כולם לקורסים חיצונים שהרי לא תמיד הם תואמים את הצורך של החברה ולכן אני לא ממהרת להוציא את האנשים לקורסים חיצוניים. ייתכן שיש כמה אנשים בתוך החברה שטובים בתחום מסוים, פייתון לדוגמה, והם יכולים להעביר את הידע לשאר העובדים. כנ"ל גם לגבי כלים טכנולוגיים. בצורה כזו חברי הצוות רוכשים ידע נרחב ומרגישים אקטואליים בשוק.

במידה וכן הוחלט להוציא כמה עובדים לקורס חיצוני, כאשר הם חוזרים הם מעבירים את הידע לשאר חברי הקבוצה. ככה כולם מרוויחים ואני יכולה להצדיק את התקציב שהוצאתי.

דבר נוסף שחשוב לי הוא העלאת היכולות הטכניות של הצוות

מה הן התובנות שלך לגבי תחום הבדיקות?

תחום הבדיקות הוא תחום רחב מאוד. הרבה אנשים רואים בו מקפצה לתחום הפיתוח ויש גם חברות שמדברות על ביטול מחלקות הבדיקות לטובת מפתחי full stack אבל בעיניי זה לא מתאים לכל אירגון ואין כמו עיניים של בודק תוכנה. התפיסה של הבודק היא תפיסה שונה מאשר תפיסתו של איש הפיתוח. אני חושבת שתחום הבדיקות ישאר איתנו עוד שנים תוך כדי התקדמות של הבדיקות הידניות לאוטומציה, משהו שכבר קורה ברוב האירגונים.

התובנה שלי היא שכולנו כאנשי מקצוע צריכים להתאים את עצמנו לכלים המשתנים ולסטנדרט שרק עולה ועולה ולא להשאר במקום גם מבחינה טכנולוגית וגם מבחינה תודולוגית.

אני חושבת שמי שנמצא בתחום צריך ללמוד כמה שיותר ולהכניס את כל הידע לצורת העבודה.

כולנו כאנשי מקצוע צריכים להתאים את עצמנו לכלים המשתנים ולסטנדרט שרק עולה ועולה ולא להשאר במקום גם מבחינה טכנולוגית וגם מבחינה מתודולוגית

מה היית ממליצה לבודק שנמצא בתחילת הקריירה?

קודם כל להתחיל. עבודה ראשונה היא לא עבודה של שכר גבוה אלא המקום לרכוש ניסיון. יש לא מעט חברות שמוכנות לקבל בודקים ללא ניסיון ואני ממליצה להפעיל את כל הקשרים שניתן כדי למצוא מקום שכזה ושם ללמוד כמה שיותר. חשוב ללמוד את התחום וללמוד את הראייה המערכתית של אותו מקום עבודה. יש הרבה מאוד בודקים שיודעים לכתוב קוד אבל לא מבינים את הראייה המערכתית ולכן לא ידעו לכתוב את הבדיקות הנכונות למוצר שעליו הם עובדים וכך באגים יכולים להתפספס בקלות. במקום העבודה הראשון חשוב להשקיע, להוכיח את עצמך וללמוד כמה שניתן. עם השנים זה כמובן נהיה קל יותר.

ספרי לנו משהו אישי עלייך

החל משנת 2012 אני כותבת בלוג מקצועי בשם Productive Hut. הבלוג התחיל בתחום של הבדיקות ועם השנים נכנסתי לתחום של ניהול זמן בצורה אפקטיבית. כך שיש לי בבלוג שילוב של שני התחומים, שילוב שהרבה פעמים אני לוקחת איתי לעבודה. חשוב לי לעמוד ביעדים עיסקיים כי אני תמיד רואה את עצמי כפרנטרית לעסק וזאת הסיבה שאני תמיד מרגישה שייכות לחברה וחושבת על העסק כעסק אישי שלי וכך קל לי יותר לגרום גם לחברי הצוות שלי להרגיש ככה. אני מאמינה שאיכות מגיעה מבפנים וניתן לראות את זה גם בלוגו של הבלוג

THETARAY

פספורט קבוצתי – ThetaRay

היא הספקית המובילה של ניתוח ביג דאטה מבוסס בינה מלאכותית - AI. החברה מסייעת לארגונים פיננסיים להילחם בפשעי סייבר פיננסיים כמו הלבנת הון, הונאה והתקפות כספומט, המשמשות למימון טרור, סמים וסחר בבני אדם, עבדות מין ומעשים זדוניים אחרים.

סוג המוצר:
Big Data מבוסס בינה מלאכותית (AI), WEB

גודל קבוצת הבדיקות:
10 אנשי בדיקות אוטומציה (קבוצת מהנדסי בדיקות בארץ וקבוצה ב-offshore).
וותק אנשי הבדיקות משתנה: החל מ-3 שנים ועד 10 שנים

מבנה הקבוצה: קבוצה מוגדרת מאנשי אוטומציה בלבד, כל אחד מהם הוא Feature Owner החל מתחילת פיתוח הפיצ'ר ועד אישורו על ידי האיש הבדיקות

סוגי בדיקות:

- רגרסיה שרצה כל לילה דרך Jenkins CI
- בדיקות שפיות
- בדיקות פונקציאונליות
- בדיקות Negative
- ולידציות
- עומסים וביצועים
- בדיקות אלגוריטמים
- בדיקות BI

כל זה חלק מהבדיקות השוטפות שלנו כחלק מהספרינט ובאחריות מלאה של הצוות.

שיטת עבודה: R&D עובד בשיטת אג'ייל, ספרינט כל שבועיים, ומקיים את כל ה-Ceremonies הרלוונטיים לאג'ייל.

צוות הבדיקות הינו חלק מסקראם הפיתוח.

בדיקות בפיתוח מונחה התנהגות
Behavior Driven Testing

פיתוח מונחה התנהגות, הינה מתודולוגיה מקובלת בפרוייקטים אג'ילייים, אך בדיקות מונחות התנהגות הן צד שאולי פחות מוכר לבודקים שבינינו ולכן בחרתי לכתוב עליהן בפינה הפעם.

נפתח בפיתוח מונחה התנהגות שבא לתאר מתודולוגיה בהנדסת תוכנה. המונח פורסם לראשונה במאמרו של דן נורת' שפורסם בשנת 2006. מאמר זה נכתב כתגובה לנושאים שעלו מפיתוח מונחה בדיקות (TDD) כמו:

1. מתי להתחיל לבדוק?
2. מה לבדוק ומה לא לבדוק?
3. כמה בדיקות לבצע בסבב אחד?
4. כיצד לכנות את שמות הבדיקה?
5. כיצד לחקור תרחיש בדיקה שנכשל?

ביסוד השיטה עומדת מחשבה מחודשת על שילוב של בדיקות יחידה שמסורתית נעשות בידי צוות הפיתוח לצד בדיקות קבלה שנעשות בדרך כלל על ידי הלקוח. מטרת השיטה הוא לחבר בעלי עניין בפרוייקט שמגיעים ללא רקע טכני ולייצר יחד תרחישי בדיקה בשפה מובנת שגם חסרי רקע בתיכנות או בדיקות ידעו להבין אותם. צוות שעובד בשיטה של BDD אמור לספק חלק ניכר מהתיעוד הפונקציונלי בצורה של סיפורי משתמש כך שכל חבר צוות יוכל לקרוא ולהבין.

סיפורי משתמש אלו נועדו לאפשר קבלת משוב מבעלי העניין שאינם טכניים לפני שלב הבדיקות. משוב זה בא לענות על השאלה החשובה: האם הפיתוח הולך לייצר את המוצר הנכון ?

כל זה אמור לקרות עוד לפני ששורת הקוד הראשונה נכתבת.

ביסודה של השיטה מתבצע הסבב הבא:

1. תיאור מדוייק של התנהגות שימוש של לקוח
2. הגדרת הדרישות מתוך סיפור המשתמש
3. הרצה של בדיקה נכשלת
4. שינוי לקוד ושיפורו
5. הרצה של בדיקה שוב בהצלחה

עקרונות השיטה:

1. שפה פשוטה ואחידה בין כל הצוותים, כזאת שכל חבר צוות גם ללא רקע טכני יוכל להבין.
2. הבדיקות נכתבות מנקודת מבט של הלקוח כאשר המיקוד הוא על הלקוח שפוגש את המוצר.

כיצד כותבים בדיקות בשיטה זאת?

הנחת העבודה בשיטה זאת היא שדרישות ובדיקות משולבות יחד, פורמט הכתיבה אחיד ויכול בקלות להפוך את הבדיקה להיות אוטומטית.

בכדי להסביר זאת אראה פה דוגמא:

הרשמה לאתר פייסבוק

Given Nitzan is on Facebook Registration page

When he enters all required registration fields

Then a Facebook account is Created

ניתן לראות שיש לנו בכל תסריט בדיקה, 3 אלמנטים:

GIVEN	(תיאור של ההקשר)
WHEN	(תיאור של הפעולה)
THEN	(תיאור של התוצאה)

כשהבדיקות הופכות מורכבות יותר וכוללות יותר אילוצים נוכל להוסיף גם משפטי AND

GIVEN	(תיאור של הקשר)
AND	(תוספת של הקשר נוסף)
WHEN	(ארוע / פעולה מתקיימים)
AND	(ארוע /פעולה נוספים מתקיימים)
THEN	(תוצאה)
AND FURTHER	(תוצאה נוספת)

ובדוגמא שלנו:

Given Nitzan is on Facebook Registration page

When he enters all required registration fields

And he hits "Join Now"

Then a Facebook account is Created

AND he is directed to the profile creation date

AND his confirmation email is sent

קובי יונסי

בעל תואר ראשון מאונ' בר אילן בתחום הלוגיסטיקה והכלכלה מוביל את לימודי בדיקות התוכנה במספר מכללות מובילות ביניהן: הטכניון – היח' ללימודי חוץ, מכללת עזריאלי להנדסה בירושלים. מייסד ושותף בהקמת QAMASTERS. מלמד אנשים כיצד לחשוב בדיקות ומסייע לארגונים לשפר מיומנויות של בודקים וללמוד כיצד לחשוב מחוץ לקופסא.

תסריטי בדיקות מונחות התנהגות ומקרי שימוש:

תסריטי בדיקות מונחי התנהגות בנויים בצורה מעט שונה מסיפורי משתמשים אך משלימים זה את זה. במקרה שימוש יש תיאור מפורט של הדרישה מנקודת מבטו של המשתמש. תרחיש בדיקות מונחי התנהגות יכלול תיאור של התנהגות המערכת שיתמוך בבדיקה.

במקרה שימוש נשתמש בפורמט:

X **AS**

Y **I Can/Want**

Z **So that**

ובמקרה של הרשמה לאתר פייסבוק:

AS a new user Nitzan

I can register a new account on the homepage

So that I can access Facebook

את המהלך הזה נוכל לכתוב כבדיקה מונחית התנהגות באופן הבא

Scenario 1: User successfully creates a Facebook Account

Given Nitzan is on Facebook Registration page

When he enters all the required registration information

And he hits "SIGN UP"

Then his Facebook account is created

AND he is directed to the profile creation page

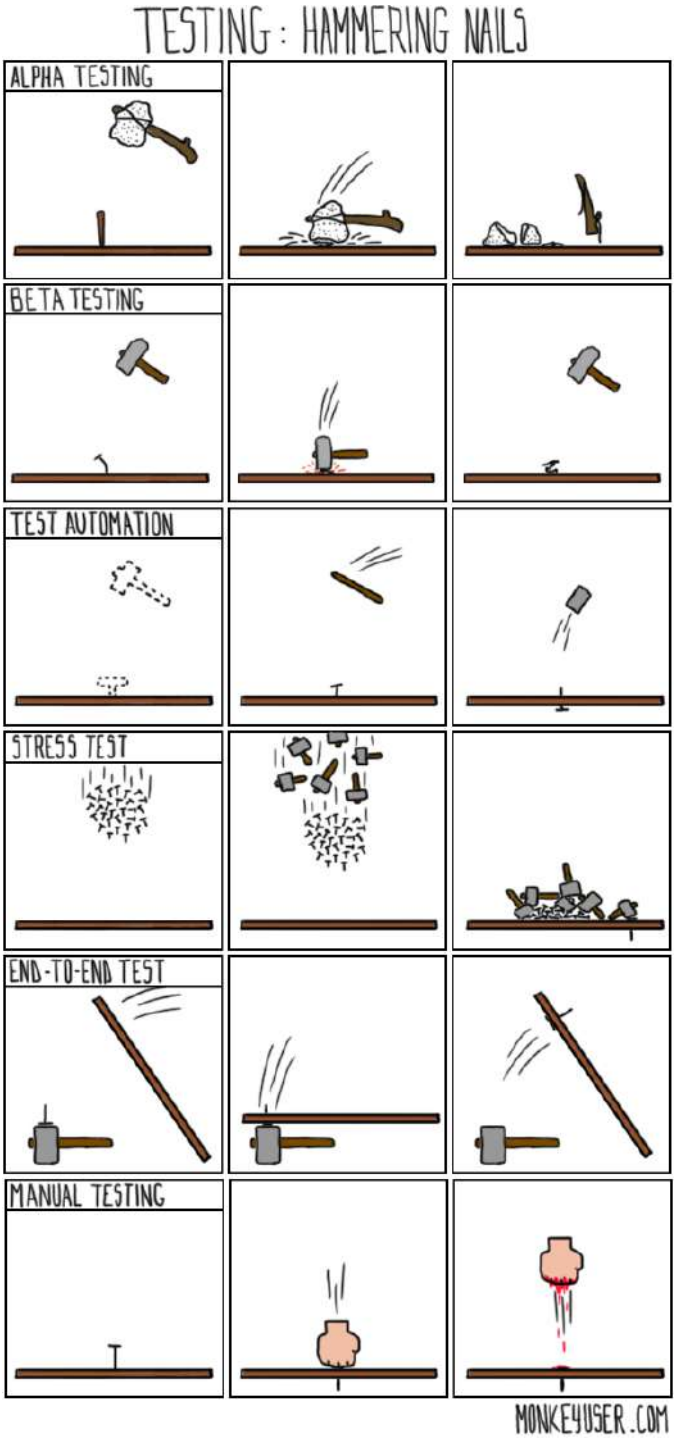
AND his confirmation email is sent

במידה ויש לנו מספר תרחישים בפונקציונליות עוקבת ניתן לצרף אותם לתרחיש זה ולייצר שרשרת של בדיקת מערכת מקצה לקצה (End to End Scenario).

לסיכום:

בדיקות מונחות התנהגות היא שיטת בדיקות שלוקחת את הבדיקות לצד של התנהגות הלקוח. שיטה זאת יכולה להתאים מאוד בפרויקטי פיתוח אג'ילי שבהם אנו מעוניינים להנגיש את הבדיקות לבעלי עניין שלא מגיעים מעולמות הטכניים. השיטה תתאים במקומות שבהם הפיתוח והבדיקות מכוונים על פי התנהגות המשתמשים. בכדי לוודא כיסוי מלא ככל שניתן לסיפורי המשתמש יש להיעזר בהיסטוריית הפעילות של המוצר ולבנות על בסיסה מגוון מצבי שימוש שבוצעו על ידי משתמשי המערכת בעבר. כך נוכל "לקלוע" באופן מדויק יותר לתהליכים אמיתיים שיפגשו את המוצר בזמן שהלקוחות ישתמשו בו כאשר יעלה לאוויר.

כמו בטכניקות אחרות, גם פה יש לדאוג להכין גם בדיקות שליליות, כלומר מצבים בהם המערכת לא אמורה לממש את הפונקציונלית שלמטרתה נועדה במידה והמשתמש מכניס ערכים שאינם ואלידיים מתוך שגיאה או בניסיון מכוון.



טיפים להכנה לראיון מקוון

ניצן גולדנברג

בשנה האחרונה בשל מצב הקורונה בארץ ובעולם יותר ויותר מעסיקים מחליטים לערוך ראיונות עבודה מקוונים באמצעות תוכנות לשיחות וידאו כגון זום, גוגל מיט ועוד, במיוחד כאשר מדובר על תפקידים המאפשרים עבודה מרחוק או ראיונות לסינון ראשוני. שיטת ראיון זו היא שונה לגמרי מראיון אישי פרונטלי מסורתי ולכן חשוב לדעת איך להתכונן ולצלוח בראיונות מסוג זה.

הנה כמה גורמים מרכזיים בראיון מקוון אשר חשוב לקחת לתשומת לב:

1 הכנות לראיון

לא משנה אם זה ראיון טלפוני, פרונטלי או מקוון, חובה עליכם להתכונן אליו מראש. חזרו על חומר הלימוד שלכם מהקורס, עברו על קורות החיים שלכם, הכינו שאלות שיכולים לשאול אתכם ותשובות אפשריות, הכינו דוגמאות ונקודות שחשוב לכם להעלות במהלך הראיון. הכינו שיעורי בית על הארגון שאתם הולכים להתראיין אליו, למדו להכיר את התוכנה איתה אתם הולכים להשתמש במהלך הראיון, בדקו את איכות הוידאו והאודיו, ביצעו שיחת בדיקה עם מישהו קרוב אליכם, שימו את הטלפון על מצב שקט.

2 סביבת הראיון

קיימו את הראיון במקום שקט ללא הפרעות, מקום שאינו ציבורי כמו בית קפה או ספריה. אם אתם גרים עם עוד אנשים בבית אז יידעו אותם מראש שיש לכם ראיון ובקשו שישמרו על שקט, סגרו חלונות באזור הראיון, השתמשו באוזניות. השתמשו ברקע ניטרלי ולא מושך את העין, לא יותר מדי מקושקש, השתמשו ברקעים שלא יעלו שאלות מיותרות מצד המראיין. עשו כל מה שנדרש כדי לארגן לכם סביבת ראיון שקטה ו-100% ריכוז.
התארגנו על חדר עם הרבה אור.

3 ישיבה

צורת הישיבה שלכם במהלך הראיון עלולה לחשוף צדדים שלכם שלא תרצו להראות כגון לחץ, ביטחון עצמי מופרז ועוד, חשוב שתשבו בצורה נינוחה אבל לא נינוחה מדי שיראה כאילו אתם שוכבים על הכיסא. הרגיעו עצמכם ואת גופכם, תהיו משוחררים ונינוחים. שבו קרוב למחשב.

4 מיקום ומצב המחשב

מקמו את המחשב במקום מרכזי על שולחן העבודה, אין לשים את המחשב על הברכיים כי לא משנה עד כמה אתם יציבים עדיין המחשב ירעד. סגרו את כל התוכנות המיותרות אשר עלולות להפריע עם נוטיפיקציות והודעות במהלך הראיון. סגרו את כל האתרים הפתוחים אצלכם במחשב ועלולים לגרום להסחות דעת, למרות שזום או תוכנות לשיחות וידאו אחרות נתמכות גם בסלולר עדיין עדיף להשתמש במחשב, המצלמה של המחשב נותנת למראיין את האפשרות לראות אתכם יותר טוב.
בדקו שאין לכם עדכוני מערכת, במידה ויש התקינו מבעוד מועד, וודאו שהמחשב שלכם בטעינה.
מצאו את רשת האינטרנט הטובה ביותר בבית. בדקו את האוזניות והרמקול של המחשב.

5 לבוש

אל תשכחו, למרות שהראיון מתקיים בבית, אתם עדיין בראיון עבודה ולכן חשוב שתתלבשו כאילו אתם יוצאים לראיון מחוץ לבית. תתארגנו כאילו יצאתם לראיון רגיל. תלבשו בגדים שתרגישו בהם נוח ויכבדו את המעמד.

6 תחילת הראיון

תפתחו את המצלמה עם חיוך קל, תציגו את עצמכם. משפט פתיחה קליל שישבור את הקרח. בדיוק כמו בתחילת פגישה פרונטלית

7 שפת גוף

במהלך הראיון שבו זקוף אבל בו זמנית גם נינוח, אל תהססו להשתמש בידיים במהלך הראיון על מנת לשמור על ערנות ועניין, תמיד שמרו על קשר עין עם המראיין, הסתכלו ישירות למצלמה ולא אל המסך או לאנשים אשר מוצגים בו.
בזמן שהמראיין מדבר הנידו בראשכם וחייכו כדי שידעו שאתם מקשיבים, המתינו שהמראיין יסיים לדבר לפני שאתם עונים והכי חשוב חייכו.

8 דפי מסרים

הכינו לצד המחשב דף עם מידע חשוב ורלוונטי שלא תרצו לשכוח כגון מילות מפתח מקורות חיים, תשובות לשאלות מאתגרות שיכולים לשאול אתכם. חשוב לא לקרוא מהדף או מהמסך אלא לענות על השאלות באופן זורם.

9 שאלו שאלות

ככל שאתם לומדים יותר על התפקיד המיועד אליו אתם מתראיינים חשבו על השאלות שאתם רוצים לשאול בסוף הראיון שלכם. הכינו את השאלות מבעוד מועד והקשיבו למראיין בזמן שהוא סוקר את החברה ואת התפקיד, בצורה זו אתם תשאלו רק שאלות בנושאים שהמראיין עדיין לא סקר. שאלות מתאימות לראיון יכולות להיות בנוגע לתרבות החברה, אחריות התפקיד המיועד או מה הכי מהנה בחברה.

והכי חשוב - שיהיה בהצלחה!

ניצן גולדנברג

מזה 6 שנים בתחום בדיקות התוכנה, תפקיד נוכחי מהנדס בדיקות בכיר בחברת SeatGeek, המוביל הראשי של קבוצת המיטאפ TestIL, מרצה בקורסים לבודקי תוכנה וחבר בצוות המייעץ AB של ITCB®.

הקופסא - תבניות חשיבה שימושיות לבדיקות
איסי חזן-פוקס

Trade-offs [1,2]

פגשתם חבר ותיק שלא ראיתם זמן רב. התעדכנתם מספר דקות כל אחד בחייו של השני עד שהחבר אומר "אני חייב ללכת עכשיו שלא אאחר" אתם נפרדים בנימוס וממשיכים לדרככם. שניכם יודעים שהוא לא באמת היה "חייב ללכת" הרי יש לו ברירה - הוא יכל לאחר לפעילות הבאה ולשאת בתוצאות, אבל שניכם מבינים שהוא התכוון לכך שהמשך השיחה שלכם יבוא **על חשבון** הגעה בזמן לפעילות הבאה והוא **בוחר** לסיים את השיחה ולא לא לאחר אליה.

קבלת החלטות טובה בכל תחום ובמיוחד בעבודה עם מערכות מורכבות מצריכה מחשבה על ה trade-off`s בין כל אפשרות - מה יבוא על חשבון מה.

כשאנו שומעים משפטים כמו "תן לנו את הפתרון הזול ביותר", "תבצעו את זה בזמן הקצר ביותר" או "אנחנו חייבים לעשות את זה בצורה הטובה ביותר", "כן בוס" היא אולי התשובה הנפוצה לבקשות כאלה אבל התשובה האמיתית היא "על מה תהיה מוכן לוותר? "You get nothing for nothing“ - לכל דבר יש עלות.

Trade-offs בתור מוצר

נסתכל למשל על אבטחה ומהירות. שתיהן תכונות חשובות במוצר שלנו, אבל פעמים רבות שיפור במדד אחד יצטרך לבוא על חשבון השני: הוספת מנגנון אבטחה עשוי להאט את הביצועים ונצטרך לקחת את זה בחשבון כשנחליט להוסיף אותו. דוגמא נוספת היא ריבוי אפשרויות מול פשטות שימוש: ככל שנוסיף אפשרויות למוצר, למשתמשים שלנו יקח יותר זמן להבין אותו והם יצטרכו לקבל יותר החלטות תוך כדי השימוש.

Trade-offs בניהול הזמן שלנו והקצאת משאבים

אנו נדרשים לקבל החלטות תכופות לגבי הקצאת הזמן שלנו למשימות שונות. כשאנו מתכננים כמה כוח אדם וכמה זמן להקצות לכל משימה, נחשוב על מה יבוא על חשבון מה.

גם אם אנחנו לא מנהלים או מובילים צוות, אנו מקבלים כל הזמן החלטות כאלה לגבי הזמן שלנו שהוא בסופו של דבר מוגבל [3].

למשל, כשאנו בודקים אזור אחד בתוכנה ורואים התנהגות חשודה שמצריכה בדיקה מעמיקה יותר, הזמן שנשקיע בבדיקה הנוספת יבוא על חשבון משימה אחרת, כמו בדיקה של אזור אחר בתוכנה שלנו. בשלב זה נבחן את ה-trade-off בין הקצאת הזמן שלנו למשימה הזו לבין המשימה האחרת. נצטרך לחשוב על ההשלכות האפשריות של הבעיה האפשרית שאנו רוצים לחקור לעומת החשיבות של בדיקת החלק האחר בתוכנה, לבצע הערכת סיכונים ולקבל החלטה.

קבלת החלטות טובה בכל תחום ובמיוחד בעבודה עם מערכות מורכבות מצריכה מחשבה על ה trade-off`s בין כל אפשרות - מה יבוא על חשבון מה.

Trade-offs בין התאמה למציאות ויכולת הסתגלות למציאות משתנה

"ככל שאתה מותאם למציאות הנוכחית כך כושר ההסתגלות שלך פוחת" [4] - נניח שסיימנו הרגע את פרוייקט האוטומציה המושלם למוצר שלנו. ככל שהפרויקט מושלם, מכסה כל פינה ומוטמע באופן חלק יותר, כך יהיה קשה יותר לעשות שינויים כדי להתאים אותו במידה והמוצר שנבדק בעזרת אותה אוטומציה השתנה. זה יהיה נכון כאשר אנחנו בונים מוצר על פי ההבנה הנוכחית שלנו, (בהקשר זה אני ממליץ למי שלא מכיר ללמוד על המושג MVP - Minimum Viable Product [5]) וגם כשיוצרים מבנה או תהליך ארגוני שמתאים בצורה אופטימלית להווה שלנו.

שימוש בתבנית חשיבה של Trade-offs מאפשר חשיבה ביקורתית על מה שאנחנו עושים וגם, לא פחות חשוב, לתקשר את ההחלטות שלנו בצורה טובה יותר. כשאנו מסבירים את השיקולים השונים וההשפעה של כל חלופה שהיתה על ההחלטה שקיבלנו, אנחנו יוצרים שקיפות ובונים אמון עם האנשים שעובדים איתנו.

לתגובות והצעות: issihf@gmail.com

איסי חזן-פוקס

המייסד הגאה של קבוצת המיטאפ TestIL, בודק תוכנה, קושחה וחומרה כבר יותר מ-20 שנה במרכז הפיתוח של אינטל בירושלים.

כיום משמש כמהנדס בדיקות מערכת בקבוצת ה Wi-Fi

1 חלק גדול מן הרעיונות המובאים כאן הינם בהשראת הפרק "Cultivating a Paradoxical Frame of Mind" בספר "The secrets of consulting" מאת ג'רי ויינברג

2 המושגים המקבילים בעברית הם "תמורות" ו"חלופות".

3 אם אין לכם אוטונומיה כזו ואתם קורבן לMicro Management כנראה שהגיע הזמן להחליף את המנהל...

4 ג'רי ויינברג, "Cultivating a Paradoxical Frame of Mind" בספרו "The secrets of consulting"

5 ראו The Lean Startup מאת Eric Reis

חושבים להחליף כלי אוטומציה הקיים אצלכם בארגון? כך תובילו תהליך POC - Proof of Concept מוצלח עם תוצאות שלא משאירות ספק!

רוצים להכניס כלי אוטומציה חדש לארגון או להחליף קיים? לא יודעים איך לגשת לנושא ואיך להתחיל תהליך POC? חוששים מתוצאות לא חד משמעיות עקב התהליך שישאירו מקום לספק?

בכתבה זו אסביר איך מובילים את התהליך בצורה שלא תשאיר מקום לספק ואיך מתאימים כלי אופטימלי עבור הארגון שלכם.

נטליה רחמני

חיה בעולם בדיקות התוכנה מעל 16 שנים ואני נושמת כל רגע את עולם האיכות.

בעלת ניסיון רב בניהול צוותים בינלאומיים, הקמת מערכי איכות, בניית מתודולוגיות בדיקות במגוון תחומים: ביטחוני, רפואי ופיננסי.

כיום Head of QA בחברת ThetaRay.

productivehut.com

החלפת הכלי או הטמעת הכלי החדש בארגון, זהו תהליך שכדאי לגשת אליו מתוכנן היטב, עם הגדרת יעדים, התהליך יכלול מה רוצים להשיג בבחירת הכלי החדש לעומת הכלי הישן. תהליך יכלול הגדרת הקריטריונים ברורים על פי מה מודדים את הכלי (קריטריונים מומלץ לאשר מראש על יד בעלי עניין הרלוונטיים. מעבר לבודקים יש צוותים נוספים שיכולים להשתמש בכלי). הצגה של קריטריונים לבעלי עניין הנוספים, יש היבט נוסף שניתן להרוויח והוא: סוג של "מחויבות" של צוותים נוספים לנושא של החלפת הכלי ושימוש עתידי בו. "מחויבות" צוותי הפיתוח יכולים לתרום לא מעט להצלחת הטמעת הכלי בארגון.

אז איך עושים את זה כהלכה למעשה?

לפני שמתחילים את התהליך חשוב מאוד להבין למה אתם רוצים לבחור כלי חדש? מה היתרונות והחסרונות המשפיעות על העבודה של הצוות ברמה היומית בכלי הקיים? תשבו עם הצוות ותבינו למה אתם רוצים להחליף את הכלי. באחד הפרוייקטים אני שמעתי לאורך הזמן מהצוות תלונות שונות על הכלי, אבל לא משהו קונקרטי ולכן קיימתי ישיבה בנושא וביקשתי להביא יתרונות וחסרונות על הכלי המשפיעות על העבודה על מנת לקבל החלטה האם ללכת לתהליך POC. ראינו צוות אשר מבזבז לא מעט זמן על תחזוקת הכלי ברמה היומית, חלק מהפונקציות לא קיימות בכלי למרות הציפייה המקצועית, כל זה לצד העלות גבוהה של הכלי. המסקנה הייתה שאנחנו רוצים ללכת לתהליך POC ולפי תוצאותיו להבין האם אנחנו מוכנים להחליף את הכלי או לא.

בואו נראה יחד מה הם השלבים הנדרשים לבחירת הכלי:

01 לאחר הבנת ה-"למה" יש לתכנן את התהליך POC כחלק ממפת הדרכים של הגרסה, מגדירים זמן סביר לביצוע POC על ידי אנשי צוות הרלוונטיים.

02 ניתן לבחור 3 כלים הקיימים בשוק להחלטה. בוא נראה איך בוחרים את שלושת הכלים לתהליך POC. סוקרים את שוק הכלים, מתייעצים עם מנהלי בדיקות אחרים ועם אנשי R&D נוספים, במילים אחרות חוקרים את "השטח" כדי להבין מה יכול להתאים לכם לארגון.

03 מגדירים מה תהליך מערכתי מקצה לקצה שרוצים לבדוק (כמובן יכול להיות יותר מאחד). בעיניי חשוב לא לבחור תהליך מורכב, אבל יש לוודא שהוא מכסה את הפונקציונליות הראשית של מערכת. המטרה כאן לראות איך הכלי מתמודד עם תהליך מקצה לקצה של המוצר.

04 להגדיר קריטריונים לבחירת כלי אוטומציה. יש לא מעט קריטריונים שניתן להתייחס אליהם כ"כלי מדידה", אבל חשוב לבחור מתוכם מה כדאי לכם ולמוצר שלכם.

ראו דוגמא:

לפני תחילת תהליך POC, אנו בוחרים קריטריונים עבור כלי אוטומציה לממשק משתמש שבחרנו לארגון שלנו. ניתן לראות שהגדרנו מה העדיפות לכל קריטריה עבור הכלי, במחשבה מה הצורך של המוצר ומה חשוב לנו בתור צוות בדיקות אוטומציה. כמובן, עדיפות וקריטריה עלולים להשתנות בהתאם לצורך ולמוצר החברה.

Criteria's	Priority	Tool #1	Tool #2
Cross Browsers Testing	High		
Supported Dev Languages	High		
Supported Test Framework	High		
Setup and Execution- CI/CD	High		
Integrations	High		
Code oriented	High-Medium		
Version Control Supported	High		
Easy maintained	High		
Debugging and logging	High		
Easy to write	High		
Reporting Capability	High		
Maturity, Documentation, Support	High		
Cost Efficient	High		
Speed	High		
Wait for Element	Medium		

Video and Screenshot support	High		
Parallel Execution	High		
Remote Execution	High		
Manage Cookies	Medium		
Repeatable, reliable	High		
Email Notifications (Custom email message received if tests are passed successfully/ failed/or any network failure)	High		
Dev Engineer skill level	High		
Installation complexity (both in time and in resources	Medium		
Load/Performance	High		

ניתן לראות שהגדרנו מגוון קריטריונים שחלקם מאוד טכניים וחלקם הקשורים לעלויות שהכלי הנבחר עלול להביא איתו, וכן אפילו כדאי להכניס קריטריה כגון מה העלות ההכשרה של הצוות בארגון. חשוב לחשוב על כל האספקטים בהטמעת הכלי שיכול להיות רלוונטי, זה יעזור לקבל תמונה מלאה ובחירה.

לאחר הגדרת הקריטריונים מומלץ לקיים פגישה עם בעלי עניין הרלוונטיים לנושא ולהציג בפניהם את הקריטריונים שנבחרו, ולהסביר מה המשמעות של כל אחד מהם.

חשוב לשמוע דעות נוספות על הקריטריונים שנבחרו, חלק מעדיפות לקריטריונים יכולה להשתנות לאחר הפגישה. דבר נוסף, להסכים על תהליך מערכתי מקצה לקצה שנבחר עבור POC. כך תהיו בטוחים שאתם בדרך הנכונה לפני תחילת ה-POC ותוכלו להרוויח "מחויבות" נוספת לתהליך מצד בעלי העניין.

05 נכנסים לתהליך POC.

שלב ראשון על פי הפרמטרים שהגדרנו אנו לומדים את הכלים, חלק מהמידע ניתן למצוא ברשת וחלק באמצעות העבודה על הכלי. (התקנה, כתיבת תרחישים, ניסיון להוציא דוחות וכו').

שלב שני הוא כתיבת תרשים מקצה לקצה בכל אחד מהכלים.

במהלך התהליך מומלץ למנהל הבדיקות לעקוב מקרוב אחרי התקדמות ה-POC ולבצע סטטוס התקדמות פעם ביומיים שלושה ולראות התקדמות לפי התכנית שהוגדרה מראש.

06 לקבוע פגישה פנים צוותית במטרה להציג התוצאות ולענות על השאלות לגבי הכלים וביצועם.

07 לקבוע פגישה מסכמת עם בעלי עניין הרלוונטיים לצורך הצגת התוצאות, שאלות וקבלת החלטה על בחירת הכלי.

לסיכום:

תהליך POC יכול להיות תהליך מאתגר, אך עם זאת תהליך מלמד, שובר שגרה והרגשה שמרימים משהו חדש עם ציפייה לתוצאות טובות יותר. חשוב להוביל תהליך בצורה מסודרת ומוגדרת מראש וכן מערבת את כל בעלי העניין על מנת להבטיח תוצאות אובייקטיביות ולא משאירות ספק.

בחן את עצמך | טל פאר

הפתרון לשאלה

תהליך הבדיקות שמציג הסילבוס מורכב מארבע רמות בדיקה (test levels):

- בדיקות רכיבים (component testing)
- בדיקות אינטגרציה (integration testing)
- בדיקות מערכת (system testing)
- בדיקות קבלה (acceptance testing)

לכל רמת בדיקה יש מטרות מוגדרות וייחודיות לאותה רמה, בסיס בדיקות (test basis) שביחס אליו מגדירים את מקרי הבדיקה (test cases), פגמים (defects) וכשלים (failures) אופייניים, וגישות ספציפיות לעיצוב וביצוע הבדיקות ברמה הזו.

כדי לייעל את העבודה בכל רמת בדיקה, על מהנדס הבדיקות להכיר את המאפיינים הנ"ל. יישום נכון של רמות הבדיקה השונות יביא לכך שפגמים (באגים) יתגלו מוקדם ובשלב המתאים ולא יתגלגלו לשלבים מאוחרים יותר, בהם הנזק גדול יותר ותיקון הבאג יהיה יקר יותר.

שאלה זו באה מפרק 2, מטרת לימוד FL-2.2.1 והיא לרמה K2 שמבקשת מהסטודנט להבין ולהסיק מסקנות.

הבה נבחן את התשובות:

א בדיקות מערכת לא בודקות את הממשקים בין רכיבים וחלקים שונים של המערכת. בדיקות אלה נעשות ברמת בדיקות האינטגרציה (integration tests).

ב שני המשפטים מתארים נכון מפרטים שעשויים לשמש כבסיס הבדיקות (test basis) עבור בדיקות רכיבים ובדיקות מערכת.

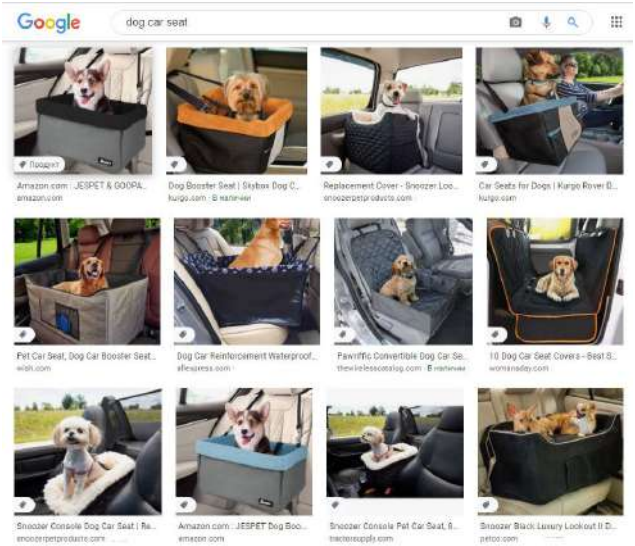
ג אמנם נהוג לחשוב על בדיקות לא פונקציונליות כבדיקות שנעשות בשלב בדיקות המערכת, אולם גם בשלבים מוקדמים יותר, כמו שלב בדיקות הרכיבים, ניתן לבצע בדיקות פונקציונליות. למשל, ניתן לתכנן ולבצע בדיקות ביצועים (performance testing) שיעריכו את צריכת משאבי המעבד לביצוע חישוב מסובך.

ד למרות שבודקים יכולים, במקרים מסוימים, לתכנן ולבצע את הבדיקות הרכיבים, הרי שבדרך כלל בדיקות אלה הן באחריות המפתחים. בדיקות המערכת הן בדרך כלל באחריות הבודקים ולא באחריות משתמשי המערכת.

לפי ההסברים הנ"ל התשובה הנכונה היא תשובה ב'. השאלה תורגמה משאלון דוגמה A של ISTQB®.
אם תרצו שאגיש שאלת דוגמה בנושא מסוים או אם יש לכם שאלות, אנא פנו אלי באימייל
tal.peer@practical-testing.com

עיקרון הקרחון

המשפחה שלי מתמחה בהמצאת רעיונות להתעשרות מהירה. לא שאנחנו מתכוונים אי פעם לעשות עם זה משהו, אבל זה שעשוע לא מזיק. כמו למשל הרעיון הנפלא לייצר כסאות בטיחות לכלבים. הרי בזמן תאונה הכלב יכול להתעופף בחלל המכונית ולפגוע באנשים! ובכלל עדיף שלא יתרוצץ וידחוף, באופן בוגדני, לשון קרה לאוזן של הנהג. רגע לפני שהשקעתי את חסכונות המשפחה בסטארט-אפ, עשיתי חיפוש מהיר בגוגל:



מאז אותו אירוע הפנמתי את מאמר קהלת: "אין חדש תחת השמש", או בשפה מודרנית יותר: תחפש בגוגל לפני שאתה מתפטר מהעבודה. זה נכון לגבי בדיקות (ראה את "כל פעם אותו סיפור", גיליון 10), וגם לגבי העיקרון החדש שהמצאתי לאחרונה: "**עקרון הקרחון**". אז זהו... שגם עקרון זה אינו חדש. הקופירייט על תאוריית הקרחון אמנם שייך לארנסט המינגווי וקשור לסגנון הכתיבה שלו, אבל זה לא הפריע לאחרים להשתמש במונח לתיאור העובדה שחלקים גדולים מכל מערכת אינם מופיעים בצורה גלויה. על אף שאינם גלויים, לחלקים אלה משמעות רבה בפעולת המערכת ואין הבנה מלאה של המערכת בלי הבנת מה ש"מתחת למים".

כיוון שרבים לפני כבר ניכסו את הביטוי של המינגווי להעברת רעיונות, אני מרגיש בסדר להצטרף לחבורה, ולהראות את הרלוונטיות של עקרון הקרחון לתחום בדיקות התוכנה.

> אנחנו בודקים את המערכת מבחוץ, ולא חשים כמה העסק מסובך בפנים

הדוגמה הפשוטה והזמינה ביותר היא היחס בין ממשק המשתמש למה שקורה "מתחתיו". קחו למשל את ממשק המשתמש של Google: מסך כמעט ריק (שדה אחד!). כל מה שהמשתמש עושה הוא להכניס כמה תווים לשדה זה. עוד לפני ההקלדה על כפתור החיפוש, המנועים של גוגל כבר בעבודה, ומציגים את החיפושים הפופולריים שמתחילים במילים שהקלדנו. כמובן שאחרי הלחיצה על "חיפוש", חוות שרתים עצומות אי שם בעולם מריצות אלגוריתמים משוכללים ומקביליים ומביאות לנו את התוצאות בפחות משנייה. וזה עובד טוב כי (בין השאר) מישהו היה מודע לחלק שהמשתמש לא רואה – החלק שמתחת למים – ובדק אותו.

אפשר לטעון שאין כאן צורך במודעות השונה ממה שנדרש בבדיקת כל מערכת אחרת. יש דרישות פונקציונליות (מה יתקבל בחיפוש) ודרישות לא פונקציונאליות (באיזה מהירות נקבל תוצאה), והבודקים צריכים לוודא אותן. אבל יש מקרים שבהם החלקים הנסתרים באמת נסתרים – בעיקר כי הם לא מיידית מתקשרים לפונקציונאליות של המערכת. אביא כאן שלוש דוגמאות.

פרוטוקול תקשורת

כמעט כל אפליקציה משתמשת בפרוטוקול תקשורת. בין אם זה לקבל מידע או שירותים מישויות אחרות או על מנת לתקשר בין חלקים שונים של אותה מערכת. בגדול, פרוטוקולים מעבירים שני סוגי מידע: (א) המטען (payload) – זה המידע שאנו רוצים להעביר (למשל: מה היתרה שלי בחשבון הבנק). (ב) מידע הנדרש על מנת שהמטען יועבר כמו שצריך ממחשב אחד לשני (למשל, כתובות).

> חלקים גדולים מכל מערכת אינם מופיעים בצורה גלויה

כבודקים, אנחנו בראש ובראשונה מתרכזים במטען. ניקח לדוגמה מסך login. נניח שאנו בודקים שרת שדורש סיסמה מהמשתמש (אתר הבנק שלנו, למשל). בבדיקות נתרכז בווידוא שמשתמשים חוקיים שמספקים סיסמה נכונה מקבלים אישור כניסה, ומשתמשים לא קיימים, או שמספקים סיסמה שגויה, לא נכנסים. יש עוד הרבה דברים אחרים סביב לזה. חלק מזה אולי אפילו מתואר בדרישות: הסיסמה צריכה להיות בעלת חוזק מינימלי; צריך לטפל בלקוח ששכח את הסיסמה; צריכים להיות מנגנונים שמאיטים את תגובת המערכת אחרי מספר ניסיונות כניסה שגויים, וכו'.

עד כאן – הכל גלוי וברור. אבל מה קורה בקרביים של הפרוטוקול? הרבה מאוד. למשל: הסיסמה עצמה לעולם לא נשלחת מהמחשב של המשתמש אל השרת. במקום זה, נשלח משהו שמוכיח שהמשתמש יודע את הסיסמה – אבל בצורה שמאזין מהצד לא יכול לזקק מתוכו את הסיסמה. שיטה מקובלת עובדת ככה: השרת שולח מספר אקראי למחשב של המשתמש. בצד של המשתמש, התוכנה מצרפת את המספר האקראי לטקסט של הסיסמה, ומכניסה את התוצאה לפונקציה חד-כיוונית (זוהי פונקציה שקשה מאוד לשחזר את הקלט שלה, בהינתן הפלט שלה; פונקציות hash הן דוגמה נפוצה). התוצאה נשלחת לשרת. השרת, שמכיר את הסיסמה של המשתמש, מבצע את אותה פעולה מתמטית, ומשווה את התוצאה המקומית שחישב למה שהמשתמש שלח. אם התוצאות זהות, הרי שהמשתמש\ת הוכיחו שהם מכירים את הסיסמה.

ראו כמה דברים עובדים כאן "מתחת למים": ייצור המספר האקראי (אם אינו ממש אקראי זה יכול להקל על פורצים לנחש סיסמאות); המימוש של הפונקציה החד-כיוונית (טעויות במימוש עלולות לפתוח פירצה); מערכת שמירת הסיסמאות בשרת ומעל הכל: הפרוטוקול עצמו שבעזרתו עוברים המסרים בין המחשבים. בעיני רוחנו השרת מקבל שתי פיסות מידע: שם וסיסמה. אבל אם נחבר רחרחן (sniffer) לקו התקשורת, יתברר לנו ששני השדות האלה נשלחים בתוך מבנה נתונים מסובך למדי: שדות שמגדירים את הפרוטוקול, את אורך המטען, כתובות, פרטים על הפרוטוקול, ועוד ועוד. לא רק זה: בדרך כלל מעורבים מספר פרוטוקולים בעניין. למשל: הודעת HTTP נשלחת כמוסה (encapsulated) בתוך פרוטוקול TCP, שכמוס בפרוטוקול IPv4, שכמוס בפרוטוקול Ethernet. לפעמים זה לא ממש מעניין – במקרים שהשרת משתמש במערכת הפעלה מלאה שמספקת את כל התשתית לטיפול בתקשורת, האפליקציה

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.

באמת מקבלת בסופו של תהליך התקשורת רק את המטען. אבל מה אם המוצר שלנו מממש שרת מינימלי במערכת משובצת, ששם כל בייט זכרון חשוב, ולכן כתבנו בעצמנו את הקוד של הפרוטוקול? עכשיו מסתבר שהפעולה הפשוטה יחסית של קבלת שני פרטי מידע ממשתמש כוללת בעצם עשרות שדות של מידע. עלינו לבדוק שהקוד בשרת מפרק (parse) נכון את התשדורות המתקבלות ומתנהג נכון בכל המקרים שתשדורות מגיעות עם פריטי מידע שגויים, לא בתחום, חסרים וכו'. הזנחת הבדיקה של חלקים אלה בקוד מעמידה את השרת בסכנה שיתמוטט בכל פעם שיגיע משהו לא מזוהה דרך הרשת. זה גם פותח פתח להתקפות זדוניות, שכן אם קלט מסויים מקריס את המערכת, קל לייצר התקפת (DOS denial of service).

> "פעולה פשוטה של קבלת שני פרטי מידע ממשתמש כוללת בעצם עשרות שדות של מידע"

כאילו שזה לא מספיק, יש עוד דברים בתחום ה"בלתי נראה". כמשתמשים, יש הרגשה שמה שאנחנו שולחים מממשק המשתמש, זה מה שמגיע לשרת. למעשה, במקרים רבים, ההודעה של המשתמש עוברת עיבוד לפני השליחה (הצפנה; או תוספת פרטים כמו שעת המשלוח). ממשק המשתמש מגן עלינו ומונע מאיתנו שליחת מידע שגוי. למשל, שעת המשלוח תהיה תמיד השעה שעל שעון המחשב, בלי יכולת לשלוט על זה. כבודקים, אנחנו צריכים להיות מודעים לתכולה המלאה של המסרים שנשלחים לשרת ולייצר מצבים פתולוגיים גם בתוספות הבלתי נראות. זה אומר שאנו לא יכולים להסתפק ביצירת בדיקות דרך ממשק המשתמש, אלא צריכים לכתוב קוד שישלח הודעות לא סטנדרטיות. כיוון בדיקות כזה נראה אולי מוגזם: למה לייצר הודעות שאין יכולת לממשק המשתמש לייצר? כיוון שמדובר בתקשורת, עלינו לקחת בחשבון שהודעות יכולות להיות מיורטות בדרך בין המשתמש לשרת, ולעבור שינוי (לא אכנס כאן לפרטים – חפשו בגוגל man in the middle attack). זהו סיכון אמיתי ולכן בבדיקת המערכת צריך לשלוח נתונים לא רק דרך ממשק המשתמש, אלא לרדת אל "מתחת למים" ולייצר הודעות שגויות וזדוניות באופן מלאכותי.

קלט של פונקציות

תיאוריית בדיקות בסיסית ממליצה לבדוק שפונקציות מבצעות בדיקות על הקלט לפני שמשתמשים בו. כך נמנע מהפעלת הפונקציה על נתונים שהם מחוץ לתחום שהפונקציה מוגדרת בו. למשל, אם הקלט לפונקציה הוא מצביע (pointer), נבדוק שהקלט אינו NULL לפני שנשתמש בו. אם מתברר שהוא כן NULL, על הפונקציה להחזיר קוד שגיאה. כל זה טוב ויפה כשמדובר בפונקציה שמקבלת קלט פשוט. אבל לעיתים, קלט לפונקציה הוא מבנה נתונים. כלומר, לא נתון אחד או שניים, אלא אולי עשר או עשרים נתונים. לעיתים קרובות חלק מהשדות במבנה הנתונים של הקלט גם הם אינם קלטים פשוטים, אלא מבני נתונים... וכו' וכו'. אחרי מעקב ופריסה של כל הקלט, יתכן שמדובר בעשרות רבות של פרטי קלט – וכל זה על פונקציה אחת! יש כאן מספר בעיות: א) בדיקות מלאות של כל השדות יוסיפו המון עבודה והמון זמן בדיקות. ב) הכנסת קוד שיבדוק את הנכונות של כל הערכים בקלט יהפוך כל פונקציה למפלצת של בדיקות קלט, וישפיע לרעה על זמן העיבוד ועל גודל התוכנה לאחר קומפילציה ג) עקרון הקרחון: לפעמים אנחנו לא מודעים כלל לעומק הסיבוך... אנחנו בודקים את המערכת מבחוץ, דרך ממשקים פשוטים יחסית (GUI או API) ולא חשים כמה העסק מסובך בפנים. גם אם היינו מודעים לכך, יהיה צורך במאמץ עצום לייצר בדיקות נגטיביות שיצליחו לחלחל קלט לא תקין לכל שדה במבנה מסובך.

מה עושים? מעבר למודעות, הערכת סיכונים וכתיבת בדיקות שמיועדות לבדוק חולשות ספציפיות, אחת הדרכים להתמודד ולמצוא באגים במקרים אלה הוא שימוש רחב ב-fuzzing. תיארתי את הטכניקה הזו במאמר בגליון הקודם (בדיקות אבטחה – גליון 24).

שמורות

כשאנחנו מריצים בדיקה, אנו מגדירים תוצאות צפויות ומוודאים בקפדנות שתוצאות אלה התקבלו. אבל מעבר לתוצאות אלה, יש המון דברים שאמורים להישאר בדיוק כמו שהיו לפני הרצת הבדיקה. אלה נקראים בעברית צחה "שמורות" (invariants). אפשר כמובן לתת אין-ספור דוגמאות: אם אני מחובר לבנק שלי דרך כרום וגם דרך פיירפוקס, הרי שיציאה מהחשבון בכרום לא אמורה להשפיע על החיבור דרך פיירפוקס. למעשה, כל מה שקורה בפיירפוקס הוא אינווריאנטי לפעולות בכרום. השמורות הן עוד דוגמה לעקרון הקרחון: הרבה פעמים אנחנו לא מודעים לצורך לחשוב על מה שלא אמור להשתנות, ומה, מכל הדברים שלא אמורים להשתנות, צריך לכסות על ידי מקרה בדיקה. אמנם אין סוף דברים שאמורים להישאר כמו שהם, אבל עם קצת מחשבה אפשר לזהות מה נמצא בסיכון לשינוי לא רצוי.

דוגמה מוכרת (ומעצבנת ביותר) היא הכנסת נתונים לטופס ברשת: אני מכניס נתון שגוי בשדה מסויים, ומנסה להתקדם למסך הבא. אני מצפה שהנתונים שהכנסתי ישמרו – גם אם הייתה טעות באחד השדות.

תוצאה צפויה: הודעת שגיאה וציון השדה הבעייתי.

שמורות: הנתונים בכל שדה שלא היה שגוי נשארים עם הערך שהוכנס.

אחרי דוגמה זאת אני מניח שלא צריך להסביר כמה החיים היו יותר טובים אם המושג של "שמורות" היה מוכר לכל הבודקים!... אלה היו רק שלוש דוגמאות. כיוון שחלקים גדולים **מכל מערכת** אינם מופיעים בצורה גלויה, סביר שגם במערכת שלכם יש לא מעט דברים מתחת למים. מתי בפעם האחרונה צללתם והסתכלתם מה יש שם?

על הקשר בין חקלאות מודרנית ובדיקות מובייל

יאיר נסימוב

סימולטורים, מכשירים פיזיים וחוות מכשירים

אם אי פעם ניסיתם להריץ סט של בדיקות מובייל, בוודאי התלבטתם האם להריץ את הבדיקות על מכשירים פיזיים או וירטואליים וכנראה שגם נתקלתם בכל הבאזז סביב חוות מכשירים. מאמר זה ינסה לעשות סדר בין הדברים ולהמליץ לכם על דרך מיטבית לפתור את סוגית השימוש בכלים השונים בבדיקות מובייל. נתחיל בסקירה של היתרונות והחסרונות של כל גישה: סימולטורים, מכשירים פיזיים וחוות מכשירים, נדון בפתרונות הפרקטיים ביותר ונקנח בחומר למחשבה עתידית בתחום.

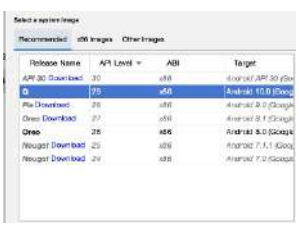
סימולטורים/אמולטורים

באופציה זו בדרך כלל בוחרים בשלבים הראשונים של פיתוח התוכנה (בהמשך המאמר תוכלו לקרוא על היתרונות והחסרונות הספציפיים של שימוש במכשירים וירטואלים). ישנם מספר הבדלים בין סימולטורים ואמולטורים, אך ההבדל אינו משמעותי לדיון במאמר זה. סימולטורים הם בדרך כלל למכשירי iOS בעוד אמולטורים נפוצים למכשירי אנדרואיד.

הטוב: כל קומבינציה של מכשיר/מערכת הפעלה נמצאת בכף ידכם (או על מסככם). אתם רוצים להריץ טסט על אייפון 7 עם iOS 12? אין בעיה, קונפיגורציה קטנה ואתם שם. כמובן שאופציה זו היא הכי זולה וזמינה. לא צריך לרכוש מכשירים חדשים או ישנים, לא צריך לתחזק אותם או לטפל בהם, לחבר אותם למחשב, להטעין אותם, לעדכן אותם ועוד.

בתמונה הבאה אפשר לראות כמה מכשירי iOS שונים זמינים לנו ברגע נתון ועוד ניתן להוריד מכשירים נוספים:

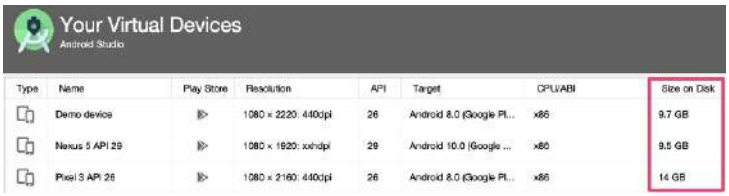
בדומה לכך, ניתן לראות עבור אנדרואיד ניתן לייצר קומבינציות רבות של מכשירים ומערכות הפעלה שונות:



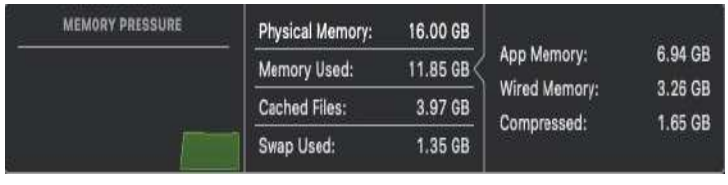
הרע: לא משנה כמה טוב הסימולטור, מכשיר מסומלץ לעולם לא יתנהג כמו מכשיר אמיתי. במיוחד אם אתם בודקי תוכנה ברור לכם שהארכיטקטורה של מכשיר נייד ומחשב שונות בתכלית (למעט שבב ה-M1 של אפל בו נדון מעט בסוף) ולמעט בדיקות פונקציונליות בתנאי מעבדה, יהיה לנו קשה עד בלתי אפשרי לבדוק דברים מורכבים יותר כמו התנהגות של מכשיר על רשת סלולרית, או במיקומים גיאוגרפיים שונים.

סימולטורים הם בדרך כלל למכשירי iOS בעוד אמולטורים נפוצים למכשירי אנדרואיד

המסובך: עבור בדיקות מכשירי iOS תזדקקו למחשב של אפל. מכשירים וירטואלים בדרך גוזלים לא מעט משאבים - מקום על הדיסק הקשיח, זכרון פנימי וכוח עיבוד לא מבוטל. לא כל מחשב יכול לעמוד בזה. יתרה מכך, המחשבים החדשים של אפל עם מעבד ה-M1 נכון להיום לא תומכים באופן מיטבי בסימולטור אנדרואיד (בניגוד למאקים מבוססי מעבדי אינטל [1]).



מחשבים אלו דווקא עושים משהו מעניין אחר. אם נסתכל על הנתונים למטה, הנה מצב המעבד שלי לפני הפעלת מכשיר אנדרואיד וירטואלי, אפשר לראות שהוא נמצא בפחות מ-10% שימוש. כמו כן הזיכרון מנוצל באופן יחסית נמוך.



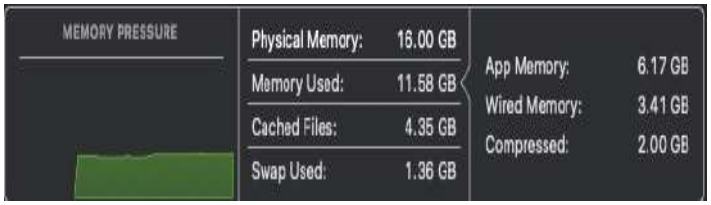
[1] https://androidstudio.googleblog.com/2020/12/android-emulator-apple-silicon-preview.html

יאיר נסימוב
בעל ניסיון של כ-10 שנים בתחום הבדיקות האוטומטיות.
מייסד חברת Rain the Dog המתמחה בייעוץ ומתן פתרונות אוטומציה מותאמים אישית לחברות.
מרצה ומפתח קורסי אוטומציה. יוטיובר מתחיל
אספן תקליטים, חובב טיולים, ריצה ואגרוף תאילנדי
נשוי + 2 + כלב - Rain.

על הקשר בין חקלאות מודרנית ובדיקות מובייל

יאיר נסימוב

כאן ניתן לראות את מצב המעבד לאחר הפעלת הסימולטור של מכשיר pixel 3. ניתן לראות שהשימוש במעבד קפץ בכ-25%. השינוי בזכרון לא נראה שהיה משמעותי לעומת זאת



מכשירים פיזיים

כמעט בכל מקום שתקראו על בדיקות מובייל תגלו שההמלצה לבדוק אפליקציות על מכשירים אמיתיים היא האופציה המועדפת. לגישה זו יש יתרונות לא מבוטלים אך גם חסרונות בולטים. תוכלו כבר לנחש **שזו האופציה היקרה ביותר והקשה ביותר לתחזוק.**

הטוב: מכשיר אמיתי הוא המוצר הסופי בו מחזיק משתמש הקצה שלכם. על כן, אין דרך טובה יותר לבדוק את האפליקציה שאתם אמונים עליה מאשר על מכשיר אמיתי. תוכלו כך לחוות כמעט כל בעיה או תקלה שיחווה משתמש אמיתי. תוכלו לבדוק את האפליקציה בתנאים שונים, כולל לצאת עם המכשיר לשטח ולבדוק אותו במיקומים גיאוגרפיים שונים, לבדוק את צריכת הסוללה, שימוש באינטרנט אלחוטי וסלולרי ועוד. בניגוד לאופציה הקודמת, לא נזדקק כלל למשאבי המחשב ותוכלו לחבר את המכשירים למכונה חלשה יחסית שרק מריצה את הבדיקות האוטומטיות על המכשירים ישירות.

הרע: כל מי שהחזיק במשרד מעבדה של מכשירים פיזיים יודע שהם נתונים לבעיות שונות. לי קרה שהגענו בבוקר למשרד ובדקנו טסט אוטומטי שנכשל באופן גורף וכשנכנסנו למעבדה גילינו שלמכשיר הסלולרי שלנו התנפחה הסוללה והתחילה לנזול החוצה. במקרים אחרים היה עדכון תוכנה כפוי שסמסונג החליטה להוציא בדיוק בלילה בו הרצנו סט בדיקות חשוב לפני שחרור גירסה ששיבש לנו את כל הריצה האוטומטית. או שמסך האייפון החליט להנעל בלילה וכל מאה הטסטים שרצו נכשלו. יש עוד דוגמאות רבות כמובן.

המסובך: השימוש במכשירים פיזיים הוא כמובן יקר, שכן כדי לכסות כמות נכבדת של בדיקות על מכשירים שונים תאלצו גם לרכוש כמות לא מבוטלת של מכשירים, דבר שעלול במהרה להפוך לעסק לא זול. יתרה מכך, זמינות של מכשירים מסוימים מוגבלת בתקופות השקה או באזורים גיאוגרפיים שונים. ישנם סיבוכים רבים נוספים כמו למשל רכישה של חבילות גלישה למכשירים או מגבלות של היצרניות (בעיקר של אפל) בהתקנה של אפליקציות בשלבי פיתוח על מכשירים פיזיים. בדרך כלל תידרשו לערוך קונפיגורציות מייגעות ולהעזר בשירותים חיצוניים דוגמת test flight ו-fire base, שגם הם כרוכים בתשלום נוסף. מי שאי פעם ניסתה להשמיש אייפון להרצה של טסטים אוטומטיים באמצעות Appium יודעת עד כמה קשה ומפרך התהליך.

מכשיר אמיתי הוא המוצר הסופי בו מחזיק משתמש הקצה שלכם. על כן, אין דרך טובה יותר לבדוק את האפליקציה שאתם אמונים עליה מאשר על מכשיר אמיתי

חוות מובייל

עיסוק בחקלאות מעולם לא היה משגשג כמו ייצור של חוות מכשירים. גישה זו באה לנסות ולפתור את הבעיות של הגישות הקודמות ולשלב ביניהם: לקחת מכשיר פיזי ולספק לנו אותו בתצורה וירטואלית. למעשה חוות המכשירים מחזיקות מכשירים פייזים ומספקות לנו גישה לדגם מסוים עם מערכת הפעלה מסוימת בתמורה לעלות חודשית לא גבוהה בדרך כלל.

הטוב: מלבד היתרונות שצוינו בפתיחה של פסקה זו נוכל למנות את העובדה שאנחנו לא זקוקים לרכישה של מכשירים רבים. כמו כן, התחזוקה והתפעול של המכשירים מנוהלים על ידי הצוות המקצועי של החברה המספקת את השירות. אין לנו כל מחויבות למכשיר ואם הוא מתקלקל נוכל לקבל גישה מיידית למכשיר אחר בעל נתונים דומים או זהים. ישנה יכולת לחלוק מכשירים בין חברי צוות שלא מצויים באותו משרד או אפילו באותה מדינה ועוד.

הרע: על כל הטוב הזה אנחנו משלמים בדרך כלל בביצועים איטיים יותר. כמו כן יש לנו הרבה פחות שליטה במכשירים. לא נוכל לקחת אותם לסיבוב בחוץ כדי לבדוק איך האפליקציה מתנהגת ברחוב עם האנטנות 5G למשל. השירות גם כרוך בתשלום קבוע והשימוש מוגבל בהתאם להסכם שלנו עם החברה.

אין לנו כל מחויבות למכשיר ואם הוא מתקלקל נוכל לקבל גישה מיידית למכשיר אחר בעל נתונים דומים או זהים

המסובך: בדרך כלל הקונפיגורציה של כתיבת טסטים אוטומטיים למכשירים הנמצאים בענן מסובכת הרבה יותר. הניטור של תקלות במכשיר קשה יותר, בעיקר כשהסשן נגמר. יתכן שהמכשיר שהוקצה לנו לבדיקה שאותה הרצנו כבר לא זמין יותר וכשנריץ שוב נקבל מכשיר שאולי מכיל את אותם נתונים אבל עלול להיות שונה פיזית.

אז מה עושים?

שילוב

בניתוח שלושת הגישות המובילות לבדיקות מובייל ראינו שאין מנצחים. בכל גישה יש יתרונות וחסרונות בולטים. אם בכל זאת נרצה המלצה אז אמליץ כאן על שילוב: בשלבים ראשוניים בלבד, בהם אנחנו מבצעים בדיקות פונקציונליות בסיסיות אין מניעה משימוש בסימולטורים. בשלב הבא, כדאי להשקיע בכל זאת במכשירים פיזיים ולהקים מעבדה מקומית. הקמת מעבדה כזו צריכה להיות מושכלת ושקולה, שלב זה יידון בהרחבה בפסקה הבאה. כמו כן, שימוש בחוות מכשירים מהווה תגבור משמעותי לכוח שלנו בכיסוי רחב של מגוון דגמים ומערכות הפעלה. על כן, הגישה הנכונה היא שילוב. כמובן שגישה זו תלוית משאבים וזמינות של העובדים בחברה שלנו.

ניהול סיכונים

במקרה של מכשירי iOS המלאכה קלה למדי שכן מרבית מכשירי ה-iOS הקיימים בשוק כבר מריצים את מערכת ההפעלה החדשה ביותר (iOS 14) - מעל ל-70 אחוזים מכלל האייפונים בשוק ומעל 80 אחוזים מהאייפונים שיוצרו ב-4 השנים האחרונות [2]. באנדרואיד המצב הרבה יותר מורכב. מרבית הטלפונים המצויים בשוק כלל לא מריצים את מערכת ההפעלה העדכנית ביותר (Android 11), למרות ההכרזות של גוגל על קצב אימוץ חסר תקדים, מדובר בפחות מ-10 אחוזים מכלל המכשירים המצויים בשוק. באנדרואיד גם קיימת הבעיה שכל יצרנית מוסיפה את השכבה שלה על מערכת ההפעלה, דבר שעלול לשנות את ההתנהגות של אפליקציות מסוימות גם על מכשירים בעלי אותה גירסה של מערכת הפעלה אך מגיעים

2 https://www.geektime.co.il/ios-14-and-android-11-adoption-rate

על הקשר בין חקלאות מודרנית ובדיקות מובייל

יאיר נסימוב

מייצרניות שונות (למשל שיאומי מול וואווי). מה שכדאי לעשות במקרה כזה הוא להשקיע בניתוח והבנה של מיהו קהל היעד שלנו, למפות את המכשירים הפופלאריים בקרב הקהל הזה ולהשיג לפחות את ה-top 5 של הרשימה הזו כמכשירים פיזיים. כמובן שנוכל להעזר בשירותי הענן בכדי לעבות את הרשימה עליה נעבור, אך בניה של שלד אנדרואידי מתאים במעבדה שלנו הוא שלב הכרחי בטרם נפנה לשירותי הענן.

סיכום ומחשבות נוספות

אם עד כה חשבנו שהאופציות היחידות שעמדו לרשותנו הן האופציות שהוצגו במאמר, כבר השנה אפל שחררה לשוק את המעבד החדש מתוצרתה - M1. מכיוון שמעבד זה הוא מבוסס ארכיטקורת ARM בו אפל משתמשת במכשירים הניידים שלה (אייפון ואייפד), הוא יכול להריץ באופן טבעי אפליקציות המיועדות למכשירים ניידים [3]. הפעילות הזו לא בשלה לחלוטין אך היא עלולה להוות תפנית מעניינת באופן בו אנחנו נבדוק את האפליקציות שלנו.

מעבר לכך, ראינו שאין גישה מנצחת, אך שאין מניעה מלהשתמש במכשירים פיזיים ולתגבר אותם בפתרונות נוספים. שירותי הענן תופסים תאוצה בשנים האחרונות והיכולות והמהירות שלהם משתנים (לטובה) כל הזמן. בסופו של יום אנחנו רוצים שהבדיקה שלנו תהיה כמה שיותר קרובה למשתמש הקצה ועלינו לעשות כל הניתן בכדי להגיע לתוצאה הזו.

3 https://www.howtogeek.com/679982/macs-will-run-iphone-and-ipad-apps-heres-how-it-will-work

לקריאה נוספת

Mobile Testing: Real Device Vs Simulator Vs Emulator Testing

https://performancelabus.com/mobile-testing-device-simulator-emulator/

Everyone talks about pros, lets talk about cons of MOBILE DEVICE FARM!!

https://www.linkedin.com/pulse/everyone-talks-pros-lets-talk-cons-mobile-device-farm-virender-singh/

Why are Device Farms so important for Software Testing?

https://www.browserstack.com/guide/importance-of-device-farms

Device Farm vs. DIY vs. Testing Platform

https://www.perfecto.io/blog/device-farm

נגיסה מ-TestIL מפגשים לבודקי תוכנה

12.01.2021

בחודש ינואר התקיים מפגש ווביר שכותרתו הייתה:

"Make it Public And other things that annoy developers about testability"

ההרצאה הועברה על ידי גיל זילברפלד, בהרצאה השתתפו אנשי בדיקות בעלי רקע ורמות ניסיון שונות.

במהלך המפגש דנו באופן שבו קוד משפיע על יכולת הבדיקה וכיצד תפקיד הבודק כולל השפעה זו.

יכולת הבדיקה חשובה מאוד במאמץ שלנו ליצור דוח מובן על מצב המערכת שלנו, ואופן כתיבת הקוד משפיע ישירות. אפילו שכפול קוד עושה את זה (גיל הראה זאת בווביר).

במידה ופספסתם את הווביר ניתן לצפות בהקלטה של הווביר ובמצגת.

ניצן גולדנברג

מזה 6 שנים בתחום בדיקות התוכנה, תפקיד נוכחי מהנדס בדיקות בכיר בחברת SeatGeek, המוביל הראשי של קבוצת המיטאפ TestIL, מרצה בקורסים לבודקי תוכנה וחבר בצוות המייעץ AB של ITCB®.

במהלך חודש ינואר מתנדבי עמותת ITCB בשיתוף עם משרד התמ"ת וקבוצת המיטאפ TestIL קיימו סדרת מפגשי ווביר שכותרתם הייתה: "הכנה לראיונות עבודה לבודקי תוכנה"

ההרצאה הראשונה והשניה הועברו ע"י ניצן גולדנברג. ניצן נתן הצצה לפעילויות של קהילת הבודקים, בנוסף ההרצאה התמקדמה ביסודות העבודה כעצמאים באתר המונים, הוצגו אילו אתרים מובילים קיימים ומה העבודה באתרי המונים יכולים לתת לבודק המתחיל מבחינת ניסיון. כמו כן, ניתנו טיפים לכתיבת קורות חיים מקצועיים, מה המשמעות בהחזקת פרופיל לינקדאין בתקופת חיפוש עבודה ולאחר מכן.

המשתתפים למדו על הראיון הטלפוני והפרונטלי / מקצועי, מה מותר ומה אסור ואיך להתכונן לראיונות ובנוסף הוצגו דוגמאות לשאלות אמיתיות מראיונות עבודה.

ההרצאה השלישית הועברה ע"י מאור שפיצר מנהל הגיוס של חברת קוואלי טסט. מאור הסביר למשתתפים איך לצלוח את סינון קורות חיים, איך להתכונן לראיון הטלפוני והפרונטלי, מה קורה בראיון המקצועי, איך לעבור על חוזה העסקה ועל מה להתמקח בשלב חתימה על החוזה ועל מה לא. בנוסף מאור נתן למשתתפים טיפים חשובים באיך להתכונן ואיך לצלוח את הראיון בזום.

במהלך הסדנה מאור קיים עם חלק מהמשתתפים סימולציות ראיון עבודה אשר במהלך הסימולציות ניתנו למשתתפים טיפים לשיפור.

נגיסה מ-TestIL מפגשים לבודקי תוכנה

ההרצאה הרביעית הועברה ע"י סלבה פשנין, ראש צוות בחברת וויקס ומנהל תחרות הבדיקות ISTC. סלבה הסביר על בדיקות פונקציונליות בסלולר, בדיקות בעזרת אנליטיקות, השוואה בין מכשירי אנדרואיד למכשירי אייפון, איך מדווחים באגים ברמה מקצועית בסלולר ואילו טרנדים יש בעולם הבדיקות. סלבה סקר שלל רחב של כלים אשר עוזרים בבדיקות.

בסוף ההרצאה המשתתפים יצאו עם צ'ק-ליסט לבדיקות סלולר.

ההרצאה החמישית הועברה ע"י קובי יונסי, מקים אתר QAMasters.co.il ומוביל תחום בדיקות תוכנה בטכניון יח' ללימודי חוץ.

קובי חשף אותנו למגמות וטרנדים הקיימים בשוק בבדיקות דפדפנים (Web), טכניקות ופרקטיקות של איך בודקים אתרי אינטרנט בתעשייה. בהרצאה חשף קובי דוגמאות לחברות מצליחות בארץ ובחו"ל ואופי הבדיקות שהן מבצעות.

לאן הולך עולם האינטרנט ואילו אתגרים חדשים עומדים בפתח.

בהמשך דיבר קובי על אופי ראיונות עבודה לבודקי Web בתחומים של פיננסיים, גיימינג, ומשרדי ממשלה. בחלק זה נחשפו המשתתפים לשאלות אמיתיות שנשאלות

בראיונות עבודה ולימד כיצד להתמודד איתם.

02.02.2021

בחודש פברואר בהמשך לסדרת הוובינרים של הכנה לראיונות עבודה התקיים מפגש וובינר בנושא "סימולציות ראיונות עבודה לבודקי תוכנה" בהנחייתו של ישראל פרכטר.

המפגש הועבר בצורת "ספיד דייטינג", ניתן היה לראות את ההתלהבות של המשתתפים והרקעים השונים מהם הגיעו. רוב הדגש המרכזי היה על איך לחבר את הסיפור האישי לפן המקצועי, ועוד קצת טיפים על דברים שכדאי להימנע מהם. בשל הכמות הגבוהה של שאלות המשתתפים הוובינר אף התארך בשעה.

עורכי הגיליון

טל פאר

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

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

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.

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.

שי ביטון

בעל 12 שנות ניסיון בפיתוח אוטומציות ובדיקות.

עובד כיום ב-Qwilt.

עמית ורטהיימר

בודק תוכנה ב-Deep Instinct.

רחל ברוך

הנדסאית תוכנה, לפני 3 שנים לאחר הפסקה של שנים מעולם ההייטק חזרה לעולם התוכנה בכל הכוח. נהנית להיות בצד הבודק עם חשיבה של מפתחת. אין יום שהיא לא לומדת משהו חדש בעולם התוכנה.

יאיר נסימוב

בעל ניסיון של כ 10 שנים בתחום בדיקות אוטומטיות.

מייסד חברת Rain the Dog המתמחה בייעוץ ומתן פתרונות אוטומציה מותאמים אישית לחברות. מרצה ומפתח קורסי אוטומציה.

יוטיובר מתחיל אספן תקליטים, חובב טיולים, ריצה ואגרוף תאילנדי נשוי + 2 + כלב - Rain

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.

תמרה מוסונובה

בודקת תוכנה ואינטגרציה בחברת Varonis.

בעלת תואר בתקשורת וקולנוע והסמכות פיתוח אפליקציות Web 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.