φιρστ παγε οφ τηε ωηολε παπερ

Κονσταντινα Σακετου

Απριλ 2022

Μέρος Ι: Πτυχιακή Εργασία

Introduction

Μέρος ΙΙ: Πρακτική Άσκηση

Εισαγωγή

Η εταιρεία στην οποία διενεργήθηκε η Πρακτική μου Άσκηση είναι η Νετσομπανψ-Ιντασοφτ. Η εταιρεία ιδρύθηκε το 1996 έχοντας την επωνυμία Ιντρασοφτ Ιντερνατιοναλ με έδρα το Λουξεμβούργο και αποτελούσε μέλος του ομίλου εταιρειών Ιντρασομ Ηολδινγς. Απο τον Οκτώβριο του 2021, ωστόσο, έγινε μέλος της Νετσομπανψ Γρουπ, η οποία αποτελεί μια από τις πιο επιτυχημένες εταιρείες πληροφορικής του Βορρά με έδρα την Δανία με έτος ίδρυσης το 2000.

Η Νετσομπανψ-Ιντρασοφτ προσφέρει λύσεις και υπηρεσίες πληροφορικής σε 500 οργανισμούς 70 χωρών παγκοσμίως. Οι φορείς αυτοί δραστηριοποιούνται σε διάφορους κλάδους, όπως αυτόν του Δημόσιου Τομέα, της Ευρωπαϊκής Ένωσης, Τραπεζικής και Χρηματοικονομικών, Κοινωνικής και Υγειονομικής Ασφάλισης, Ενέργειας, Τηλεπικοινωνιών κ.ά. Η επιχείρηση απασχολεί περίπου 2800 εργαζόμενους 50 διαφορετικών εθνικοτήτων, το οποίο επιτυγχάνει διατηρώντας γραφεία σε 13 χώρες. Έτσι, εκδηλώνεται το ενδιαφέρον της εταιρίας για διατήρηση και υιοθέτηση της πολυπολιτισμικότητας και διαφορετικότητας στο περιβάλλον της, επιτυγχάνοντας με αυτόν τον τρόπο την ανταλλαγή διαφορετικών απόψεων και τη δημιουργική συνύπαρξη των ανθρώπων της σε ένα ενιαίο πλάισιο.

Η εταιρεία με τοποθέτησε στο κομμάτι που παρέχει λύσεις στην Ευρωπαϊκή Επιτροπή (Ευροπεαν ὃμμισσιον) και ,συγκεκριμένα, σε προθεστς που αφορούν προϊόντα της Ευρωπαϊκής Στατιστικής Υπηρεσίας (Ευροστατ). Καθήκον της υπηρεσίας αυτής αποτελεί η συλλογή και δημοσίευση στατιστικών δεδομένων και μελετών που αφορούν τις χώρες της Ευρωπαϊκής Ένωσης. Οι ωεβ εφαρμογές του οργανισμού αυτού εξυπηρετούν στην διαχείριση και ανταλλαγή στατιστικών δεδομένων μεταξύ των χωρών.

Κατά την διάρκεια της πρακτικής άσκησης, απασχολήθηκα στο κομμάτι του Αππλιςατιον Τεστινγ. Συγκεκριμένα, εντάχθηκα στο Σοφτωαρε Τεστινγ Σεριςες ἔντερ - ΣΤΣ", το οποίο αποτελεί ένα από τα μεγαλύτερα αυτόνομα Τεστ ἔντερς με πάνω από 100 Τεστ Αναλψστς και Τεστ Ενγινεερς. Θέλοντας να ασχοληθώ με το κομμάτι του Τεστ Δεελοπμεντ, απέκτησα τον ρόλο του Τεστ Αυτοματιον Ενγινεερ. Απασχολήθηκα, δηλαδή, σε οτιδήποτε αφορά τεστινγ λογισμικού μέσω αυτοματοποιημένων τεστ σρριπτς.

Χαρακτηριστικά Πρακτικής Άσκησης

Software Testing Services Center

Το Software Testing Services Center στο οποίο εντάχθηκα σκοπεύει στην διασφάλιση της ποιότητας των προϊόντων της εταιρείας αλλά και των πελατών της. Είναι υπεύθυνο για τον έλεγχο του εάν οι εφαρμογές λειτουργούν με τον αναμενόμενο και πιο αποδοτικό τρόπο και αυτό επιτυγχάνεται μέσω της εφαρμογής πολυάριθμων ειδών ελέγχων στα αντίστοιχα λογισμικά. Τα είδη των ελέγχων αυτών παραθέτονται παρακάτω:

Performance Testing
Σκοπός του συγκεκριμένου είδους ελέγχου είναι να εξασφαλίσει την ταχύτητα, επεκτασιμότητα και σταθερότητα του λογισμικού. Συγκεκριμένα, δίνει έμφαση σε μετρικές όπως ο χρόνος απόκρισης της εφαρμογής, η αξιοπιστία, ο τρόπος με τον οποίο χρησιμοποιούνται οι πόροι αποσκοπώντας στην όσο το δυνατόν πιο αποδοτική τους αξιοποίηση. Το Performance Testing αποτελείται από τα τρία παρακάτω είδη ελέγχων.

- Load Testing
  Στόχος του Load Testing είναι να ελέγξει το πώς ανταποκρίνεται μια εφαρμογή, ένα σύστημα σε μεγάλο αριθμό ταυτόχρονων χρηστών, οι οποίοι είναι ενεργοι και εκτελούν διάφορες ενέργειες σε αυτό.

- Volume Testing
  Όταν εφαρμόζουμε Volume Testing, παρέχουμε στην εφαρμογή μας ένα πολύ μεγάλο όγκο δεδομένων. Στόχος είναι να διαπιστωθεί έαν το σύστημα λειτουργεί όπως αναμένεται, ακόμη και με έναν μεγάλο όγκο δεδομένων. Παρά το γεγονός ότι αυτό το είδος ελέγχου μοιάζει με το Load Testing, όταν εφαρμόζουμε το δεύτερο, θέλουμε να δούμε το πώς μεταβάλλεται και επηρεάζεται η απόδοση της εφαρμογής μας όταν αυξάνεται ο όγκος δεδομένων, χωρίς να δίνουμε βάση στον τρόπο εκτέλεσης των αναμενόμενων λειτουργιών.

- Stress Testing
  Στόχος του Stress Testing είναι να εντοπίσει το ανώτατο όριο φόρτου (χρηστών, πόρων κλπ.) στο οποίο η εφαρμογή μπορεί να ανταποκριθεί αποτελεσματικά. Ακόμη, βάση δίνεται και στο πόσο ομαλή θα είναι η επαναφορά του συστήματος στο αναμενόμενο φόρτο πόρων μετά από ένα πιο απαιτητικό χρονικό διάστημα όσον αφορά την απαιτούμενη απόκριση.

GUI Testing
Σε αυτό το είδος ελέγχου, στόχος είναι να διασφαλίσει ότι η γραφική διεπαφή του χρήστη είναι σχεδιασμένη σύμφωνα με τις απαιτήσεις και οδηγίες του πελάτη και πως μέσω αυτής

εκτελούνται αποτελεσματικά όλες οι λειτουργίες της εφαρμογής. Συγκεκριμένα, ελέγχεται η εμφάνιση των οθονών, τα διάφορα κουμπιά, τα πεδία εισόδου δεδομένων (checkboxes, radio buttons, text input fields κλπ.), καθώς επίσης και η λειτουργικότητά τους.

API Testing
Κατά το συγκεκριμένο είδος ελέγχου, διασφαλίζονται διάφορες μετρικές του συστήματός μας όπως απόδοση, λειτουργικότητα κλπ. Τα APIs δεν περιέχουν διεπαφή χρήστη, οπότε όλος ο έλεγχος γίνεται σε επίπεδο διαδικτυακών συναλλαγών μηνυμάτων (web transactions) μεταξύ web client και web server. Γενικότερα το API Testing εξυπηρετεί την μεθοδολογία Agile Software Development, καθώς προσαρμόζεται εύκολα σε νέα λειτουργικότητα, σε αντίθεση με τους ελέγχους μέσω γραφικής διεπαφής, οι οποίοι απαιτούν περισσότερο χρόνο για να συντηρηθούν.

Data Migration Testing
Αυτός ο τύπος ελέγχου εφαρμόζεται στην περίπτωση που μια εφαρμογή ή ένα σύστημα πρόκειται να μεταφερθεί σε μια νέα υποδομή. Κατά το Migration Testing, διασφαλίζεται πως η εφαρμογή μεταβαίνει στη νέα υποδομή χωρίς να υπάρχει κάποια συνέπεια στην ακεραιότητα των δεδομένων. Επίσης, επιβεβαιώνεται πως δεν υπήρξε απώλεια δεδομένων κατά τη μεταφορά.

Θέση Πρακτικής Άσκησης

Όπως αναφέρθηκε και παραπάνω, το Software Testing Services Center αποτελείται από Test Analysts και Test Engineers. Εγώ ακολούθησα το μονοπάτι του Test Engineer, λαμβάνοντας συγκεκριμένα τον ρόλο του Test Automation Engineer, του οποίου τα καθήκοντα περιγράφονται παρακάτω.

- Ανάλυση των σεναρίων ελέγχου, μελέτη και διερεύνηση των λειτουργικών και μη λειτουργικών απαιτήσεων του συστήματος που πρόκειται να υποβληθούν σε έλεγχο.
- Ανάλυση του τι αξίζει να αυτοματοποιηθεί, μελέτη των σεναρίων ελέγχου και καθορισμός αυτών που χρίζουν αυτοματοποίησης. Τα υπόλοιπα σενάρια ελέγχου εκτελούνται manually από τους Manual Testers.
- Σχεδιασμός και προετοιμασία των αυτοματοποιημένων ελέγχων και δημιουργία των απαραίτητων δεδομένων ελέγχου.
- Οργάνωση των test cycles, εκτίμηση του απαιτούμενου χρόνου υλοποίησης και ολοκλήρωσης των καθορισμένων ελέγχων για να παραδοθεί έγκαιρα η αντίστοιχη αναφορά στους stakeholders.
- Προετοιμασία των στατιστικών δεδομένων από την εκτέλεση των αυτοματοποιημένων ελέγχων της εφαρμογής. Στη συνέχεια, τα δεδομένα αυτά ενσωματόνονται και υποβάλονται στους stakeholders.

Μεθοδολογίες και Εργαλεία

Η ομάδα του Software Testing Services Center χρησιμοποιεί έναν μεγάλο αριθμό εργαλείων, μεθοδολογιών και τεχνολογιών για την εκτέλεση των λειτουργιών του. Σε αυτό το μέρος γίνεται μια αναφορά σε όλα τα παραπάνω.

Μεθοδολογίες

Behavior Driven Development
κείμενο

Page Object Pattern
κείμενο

Εργαλεία

Java
Η γλώσσα προγραμματισμού που χρησιμοποιείται σε αρκετά μεγάλο βαθμό για τον έλεγχο των συστημάτων και των εφαρμογών είναι η Java. Αποτελεί μια αντικειμενοστραφής γλώσσα, βασίζεται, δηλαδή, σε κλάσεις και αντικείμενα. Χάρη στις πολυάριθμες δυνατότητες και εφαρμογές της, δίνει την δυνατότητα εκτέλεσης πολλών περίπλοκων λειτουργιών και ανταποκρίνεται, έτσι, στις απαιτήσεις του τμήματος.

Maven
Το Maven αποτελεί ένα εργαλείο αυτοματοποίησης συνήθως διαφόρων projects γραμμένων σε Java. Εξυπηρετεί στην οργάνωση και την αποτελεσματικότερη διαχείριση της δομής τους. Ακόμη, διευκολύνει διάφορα στάδια του Development Life Cycle με το testing να είναι ένα από αυτά.

Git
Το Git αποτελεί ένα Version Control Tool, το οποίο επιτρέπει την ευκολότερη διαχείριση διαφόρων σταδίων ανάπτυξης ενός project. Εξυπηρετεί στον ευκολότερο εντοπισμό των αλλαγών που συμβαίνουν, διευκολύνοντας την ομαδική συνεργασία και ανάπτυξη κώδικα. Η ομάδα χρησιμοποιεί το GitLab για την διαχείριση των repositories.

Selenium
Το εργαλείο αυτό είναι ένα από τα πιο βασικά εργαλεία που χρησιμοποιεί η ομάδα. Προσφέρει αυτοματοποίηση των φυλλομετρητών και χρησιμποιείται έντονα στον τομέα του Automation Testing. Μέσω των βιβλιοθηκών που παρέχει, επιτυγχάνεται η ευκολότερη και γρήγορη εκτέλεση αυτοματοποιημένων ελέγχων.

Cucumber
Το Cucumber αποτελεί ένα εργαλείο που υποστηρίζει το Behavior Driven Development - BDD που αναφέρθηκε και παραπάνω. Χάρη στη γλώσσα Gherkin, η αναμενόμενη συμπεριφορά του λογισμικού εκφράζεται με πολύ απλό και κατανοητό τρόπο, ο οποίος πλησιάζει αρκετά την

φυσική γλώσσα. Έτσι, γίνεται εύκολα κατανοητή από τον οποιονδήποτε και, συνεπώς, από τους πελάτες.

REST-Assured API
Το συγκεκριμένο εργαλείο παρέχεται από την Java και εξυπηρετεί στο αυτοματοποιημένο API Testing. Συγκεκριμένα, εξυπηρετεί στον έλεγχο των REST Services, μέσω HTTP Requests.

SoapUI
Το SoapUI αποτελεί και αυτό ένα εργαλείο για Web Service Testing. Παρ΄ όλα αυτά, μπορεί να χρησιμοποιηθεί και για Functional και Load Testing.

JMeter
Το JMeter αποτελεί ένα εργαλείο, το οποίο χρησιμοποιείται για Performance Testing. Παρέχει, δηλαδή, στην εφαρμογή μας έναν μεγάλο όγκο δεδομένων με στόχο την καταγραφή του πώς αυτό ανταποκρίνεται και το πώς αποδίδει.

?bibname?

[1] Danielle Gonzalez, Suzanne Prentice, and Mehdi Mirakhorli. A fine-grained approach for automated conversion of junit assertions to english. In *Proceedings of the 4th ACM SIGSOFT International Workshop on NLP for Software Engineering*, NL4SE 2018, page 14–17, New York, NY, USA, 2018. Association for Computing Machinery.

[2] Dr Srinivas Perala and Ajay Roy. A review on test automation for test cases generation using nlp techniques. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, 12(6):1488–1491, 2021.

[3] Man Zhang, Tao Yue, Shaukat Ali, Huihui Zhang, and Ji Wu. A systematic approach to automatically derive test cases from use cases specified in restricted natural languages. In *International Conference on System Analysis and Modeling*, pages 142–157. Springer, 2014.

[4] Dionny Santiago, Peter J. Clarke, Patrick Alt, and Tariq M. King. *Abstract Flow Learning for Web Application Test Generation*, page 49–55. Association for Computing Machinery, New York, NY, USA, 2018.

[5] Dionny Santiago, Tariq M King, and P Clarke. Ai-driven test generation: machines learning from human testers. In *Proceedings of the 36th Pacific NW Software Quality Conference*, pages 1–14, 2018.

[6] Xuan Phu Mai, Fabrizio Pastore, Arda Gǿknil, and Lionel Briand. A natural language programming approach for requirements-based security testing. In *29th IEEE International Symposium on Software Reliability Engineering (ISSRE 2018)*. IEEE, 2018.

[7] Mathias Soeken, Robert Wille, and Rolf Drechsler. Assisted behavior driven development using natural language processing. In *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 269–287. Springer, 2012.

[8] Chunhui Wang, Fabrizio Pastore, Arda Goknil, and Lionel Briand. Automatic generation of acceptance test cases from use case specifications: an nlp-based approach. *IEEE Transactions on Software Engineering*, 2020.

[9] Prerana Pradeepkumar Rane. *Automatic generation of test cases for agile using natural language processing*. PhD thesis, Virginia Tech, 2017.

[10] Sunil Kamalakar, Stephen H Edwards, and Tung M Dao. Automatically generating tests from natural language descriptions of software behavior. In *ENASE*, pages 238–245, 2013.

[11] Filipe Arruda, Flávia Barros, and Augusto Sampaio. Automation and consistency analysis of test cases written in natural language: An industrial context. *Science of Computer Programming*, 189:102377, 2020.

[12] Sahar Tahvili, Leo Hatvani, Michael Felderer, Wasif Afzal, Mehrdad Saadatmand, and Markus Bohlin. Cluster-based test scheduling strategies using semantic relationships between test specifications. In *Proceedings of the 5th International Workshop on Requirements Engineering and Testing*, RET '18, page 1–4, New York, NY, USA, 2018. Association for Computing Machinery.

[13] Linyi Li, Zhenwen Li, Weijie Zhang, Jun Zhou, Pengcheng Wang, Jing Wu, Guanghua He, Xia Zeng, Yuetang Deng, and Tao Xie. Clustering test steps in natural language toward automating test automation. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2020, page 1285–1295, New York, NY, USA, 2020. Association for Computing Machinery.

[14] Edmund Wong, Lei Zhang, Song Wang, Taiyue Liu, and Lin Tan. Dase: Document-assisted symbolic execution for improving automated software testing. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 1, pages 620–631. IEEE, 2015.

[15] Abhishek Sainani, Preethu Rose, Vivek Joshi, and Smita Ghaisas. Extracting and classifying requirements from software engineering contracts. pages 147–157, 08 2020.

[16] Wolfgang Mueller, Alexander Bol, Alexander Krupp, and Ola Lundkvist. Generation of executable testbenches from natural language requirement specifications for embedded real-time systems. In *Distributed, Parallel and Biologically Inspired Systems*, pages 78–89. Springer, 2010.

[17] Tao Yue, Shaukat Ali, and Man Zhang. Rtcm: a natural language based, automated, and practical test case generation framework. In *Proceedings of the 2015 international symposium on software testing and analysis*, pages 397–408, 2015.

[18] Satoshi Masuda, Tohru Matsuodani, and Kazuhiko Tsuda. Syntactic rules of extracting test cases from software requirements. In *Proceedings of the 2016 8th International Conference on Information Management and Engineering*, pages 12–17, 2016.

[19] Harbhajan Singh, Mirko Conrad, and Sadegh Sadeghipour. Test case design based on z and the classification-tree method. In *First IEEE International Conference on Formal Engineering Methods*, pages 81–90. IEEE, 1997.

[20] Gustavo Carvalho, Diogo Falcao, Flávia Barros, Augusto Sampaio, Alexandre Mota, Leonardo Motta, and Mark Blackburn. Nat2testscr: Test case generation from natural language requirements based on scr specifications. *Science of Computer Programming*, 95:275–297, 2014.

[21] Xiaomeng Chen, Jinbo Wang, Shan Zhou, Panpan Xue, and Jiao Jia. Test case reuse based on esim model. In *2021 8th International Conference on Dependable Systems and Their Applications (DSA)*, pages 700–705. IEEE, 2021.

[22] Edgar Sarmiento, Julio CSP Leite, Eduardo Almentero, and Guina Sotomayor Alzamora. Test scenario generation from natural language requirements descriptions based on petrinets. *Electronic Notes in Theoretical Computer Science*, 329:123–148, 2016.

[23] Avik Sinha, Stanley M Sutton, and Amit Paradkar. Text2test: Automated inspection of natural language use cases. In *2010 Third International Conference on Software Testing, Verification and Validation*, pages 155–164. IEEE, 2010.

[24] Benedikt Eberhardinger, Axel Habermaier, and Wolfgang Reif. Toward adaptive, self-aware test automation. In *2017 IEEE/ACM 12th International Workshop on Automation of Software Testing (AST)*, pages 34–37. IEEE, 2017.

[25] Pablo Pedemonte, Jalal Mahmud, and Tessa Lau. Towards automatic functional test execution. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, pages 227–236, 2012.

[26] Jane Huffman Hayes, Alex Dekhtyar, and Senthil Karthikeyan Sundaram. Advancing candidate link generation for requirements tracing: The study of methods. *IEEE Transactions on Software Engineering*, 32(1):4–19, 2006.

[27] JJ Gutiérrez, MJ Escalona, and M Mejías. A model-driven approach for functional test case generation. *Journal of Systems and Software*, 109:214–228, 2015.

[28] Liping Zhao, Waad Alhoshan, Alessio Ferrari, Keletso J Letsholo, Muideen A Ajagbe, Erol-Valeriu Chioasca, and Riza T Batista-Navarro. Natural language processing for requirements engineering: A systematic mapping study. *ACM Computing Surveys (CSUR)*, 54(3):1–41, 2021.

[29] Vahid Garousi, Sara Bauer, and Michael Felderer. Nlp-assisted software testing: A systematic mapping of the literature. *Information and Software Technology*, 126:106321, 2020.