ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ
ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΥΤΟΜΑΤΟΠΟΙΗΣΗ ΔΙΑΔΙΚΑΣΙΑΣ ΔΟΚΙΜΩΝ ΜΕ ΧΡΗΣΗ ΕΤΟΙΜΩΝ ΠΡΟΤΥΠΩΝ

Χριστόδουλος Πάσιος

Βασίλης Τουρκαντώνης

ΕΠΙΒΛΕΠΩΝ: Σωτήρης Π. Χριστοδούλου, Επίκουρος καθηγητής

ΠΑΤΡΑ 2023

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Πάτρα, Ημερομηνία

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Ονοματεπώνυμο, Υπογραφή

2. Ονοματεπώνυμο, Υπογραφή

3. Ονοματεπώνυμο, Υπογραφή

Υπεύθυνη Δήλωση Φοιτητών

Βεβαιώνουμε ότι είμαστε συγγραφείς αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχαμε για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης έχουμε αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνουμε ότι αυτή η εργασία προετοιμάστηκε από εμάς προσωπικά ειδικά για τη συγκεκριμένη εργασία. Η έγκριση της διπλωματικής εργασίας από το Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Πελοποννήσου δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα εκ μέρους του Τμήματος. Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία των φοιτητών Χριστόδουλο Πάσιο και Βασίλη Τουρκαντώνη που την εκπόνησαν. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης ο συγγραφέας/δημιουργός εκχωρεί στο Πανεπιστήμιο Πελοποννήσου, μη αποκλειστική άδεια χρήσης του δικαιώματος αναπαραγωγής, προσαρμογής, δημόσιου δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσής τους διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος και για όλο το χρόνο διάρκειας των δικαιωμάτων πνευματικής ιδιοκτησίας. Η ανοικτή πρόσβαση στο πλήρες κείμενο για μελέτη και ανάγνωση δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, αποθήκευση, πώληση, εμπορική χρήση, μετάδοση, διανομή, έκδοση, εκτέλεση, «μεταφόρτωση» (downloading), «ανάρτηση» (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού. Ο συγγραφέας/δημιουργός διατηρεί το σύνολο των ηθικών και περιουσιακών του δικαιωμάτων.

Περίληψη

Στην εν λόγω πτυχιακή εργασία θα μελετηθεί η ανάπτυξη ενός μηχανισμού αναγνώρισης κρυμμένων αντικειμένων του DOM (Document Object Model), με σκοπό τη χρήση του σε ένα περιβάλλον δοκιμών λογισμικού, χρησιμοποιώντας λύσεις ανοιχτού λογισμικού.

Ο σκοπός αυτός επιτυγχάνεται μέσω της εξαγωγής προτύπων, από στιγμιότυπα εικόνων του προγράμματος περιήγησης, και τη μετέπειτα χρήση τους σε μία διαδικασία αναγνώρισης εικόνας. Μέσω αυτής της επιλογής, παρέχεται στο εργαλείο δοκιμών η δυνατότητα να αντιλαμβάνεται σε πλήρη έκταση την κατάσταση της ιστοσελίδας, επιτρέποντας του να ενεργεί σε οποιοδήποτε σημείο πάνω σε αυτή.

Τα κύρια εργαλεία για την ανάπτυξη του εργαλείου δοκιμών είναι το Selenium Framework και η βιβλιοθήκη OpenCvSharp. Για την ανάπτυξη του περιβάλλοντος δοκιμών, χρησιμοποιήθηκαν οι τεχνολογίες HTML, CSS, Bootstrap και JavaScript. Ο κώδικας του παιχνιδιού θα φιλοξενηθεί στο αποθετήριο κώδικα GitHub, ενώ το παιχνίδι θα είναι προσβάσιμο μέσω της υπηρεσίας GitHub Pages.

Abstract

The aim of this thesis is the development of a mechanism for identifying hidden objects of the DOM (Document Object Model), to use it in a software testing project. To achieve this purpose, we will use open-source tools and libraries.

This cause can be achieved by extracting image templates from browser snapshots and then using them in an image recognition process. Through this process, the testing framework can recognize the web application and act on it.

The main tools for developing the testing framework are Selenium and OpenCV library. In addition, a demo game will be developed and will be used as the testing environment. CSS, Bootstrap and JavaScript language have been used for the development of that game. The game code will be hosted in a public repository on GitHub, while the game will be accessible through the GitHub Pages service.

Πίνακας Συντομογραφιών

Συντομογραφία	Έννοια
HTML	Hyper Text Markup Language
JS	Javascript
CSS	Cascading Style Sheet
DOM	Document Object Model
XPaths	Extensible Markup Language Paths
HTTP	Hypertext Transfer Protocol
CDN	Content Delivery Network
URL	Uniform Resource Locator
CLI	Command Line Interface
JSON	JavaScript Object Notation
CIL	Common Intermediate Language
CLR	Common Language Runtime
DLL	Dynamic Link Library
IDEs	Integrated Development Environment
RC	Remote control
GPU	Graphics Processing Unit

Περιεχόμενα

1. Εισαγωγή

Τις τελευταίες δεκαετίες οι δοκιμές αποτελούν αναπόσπαστο κομμάτι της παραγωγής και της διάθεσης ενός προϊόντος λογισμικού σε ένα εμπορικό και μη περιβάλλον. Αυτό συμβαίνει, διότι οι οργανισμοί στην προσπάθεια τους να καλύψουν τις ανάγκες των τελικών χρηστών και των πελατών τους, καλούνται να αναπτύξουν ή να συντηρήσουν μεγάλες εφαρμογές με αυξημένη πολυπλοκότητα. Η ανάπτυξη τέτοιων εφαρμογών, έχει οδηγήσει στην ανάγκη ύπαρξης τμημάτων διασφάλισης ποιότητας λογισμικού.

Μέχρι πρότινος, τα τμήματα αυτά επανδρώνονταν με προσωπικό, το οποίο είχε την ευθύνη της ανάλυσης των τεχνικών προδιαγραφών του συστήματος, ως στόχο τη δημιουργία επαρκών δοκιμών λογισμικού, οι οποίες εκτελούνταν μόνο χειροκίνητα (Manual Testing). Πολλές φορές όμως, η μη ύπαρξη τμημάτων διασφάλισης λογισμικού ή η απουσία επαρκούς αριθμού ατόμων, οδηγούσε τους οργανισμούς σε συνεργασίες με τρίτες εταιρείες, οι οποίες κατείχαν την τεχνογνωσία και παρείχαν καταρτισμένο προσωπικό.

Η επιλογή αυτή φαντάζει συνετή, εάν αναλογιστεί κανείς το μέγεθος των προβλημάτων που θα προέκυπταν από την πλήρη απουσία της διαδικασίας της ανάλυσης και της ορθής εκτέλεσης δοκιμών, αλλά και τον αντίκτυπο που θα είχε αυτή στο κύρος ενός οργανισμού. Ωστόσο, τέτοιου είδους συνεργασίες είναι ικανές να αυξήσουν σημαντικά το κόστος κατά την υλοποίηση και τη συντήρηση του προϊόντος.

Ακόμα όμως και με την ύπαρξη τμημάτων εντός των οργανισμών, για την εκτέλεση χειροκίνητων δοκιμών, το κόστος παραμένει σε αρκετά υψηλά επίπεδα. Ταυτόχρονα, χρειάζεται να δαπανηθεί αρκετός χρόνος για την εκτέλεση επαναλαμβανόμενων δοκιμών που σκοπό έχουν τη διασφάλιση ποιότητας των υπαρχόντων λειτουργικοτήτων της εφαρμογής, σε συνδυασμό με την προσθήκη δοκιμών, για τις επιπλέον λειτουργίες των νέων εκδόσεων.

Πολλές εταιρείες και οργανισμοί προσαρμόζουν τη μεθοδολογία παραγωγής λογισμικού (Software Development Life Cycle) που ακολουθούν, και εντάσσουν μέσα σε αυτή, την αυτοματοποίηση των δοκιμών (Automation Testing). Υπάρχουν αρκετοί τύποι δοκιμών, όπου ο κάθε ένας επιλέγεται με βάση τη δοκιμαστική προσέγγιση που επιλέγει το αντίστοιχο τμήμα διασφάλισης ποιότητας της εταιρείας ή του οργανισμού. Ανάλογα το μοντέλο του κύκλου ζωής του λογισμικού, που έχει επιλεγεί από τον αντίστοιχο οργανισμό, όταν η φάση του

προγραμματισμού τελειώσει ξεκινάει η φάση των δοκιμών του. Το στάδιο αυτό κατατάσσεται αρκετές φορές σαν η πιο κοστοβόρα φάση του κύκλου ζωής.

Η επιλογή τους για αυτοματοποίηση των δοκιμών συνάδει με την προσπάθεια μείωσης του χρόνου εκτέλεσης των δοκιμών, χωρίς όμως να μειωθεί ο ζητούμενος αριθμός δοκιμών που πρέπει να εκτελεστεί για να διασφαλιστεί η ποιότητα του παραδοτέου λογισμικού. Ο αριθμός των συνολικών δοκιμών μπορεί να αυξηθεί δραματικά, και οι αυτοματοποιημένες δοκιμές μπορούν επίσης να εξασφαλίσουν περισσότερους ανθρώπινους πόρους, για διερευνητικές δοκιμές σε νέες προσθήκες του λογισμικού.

Υπάρχουν αρκετά εργαλεία ανοικτού κώδικα και μη, που μπορούν να χρησιμοποιηθούν από τον μηχανικό για να για να επιτευχθεί η αυτοματοποίηση εφαρμογών ιστού. Αυτά μπορούν να διαχωριστούν σε κατηγορίες, όπως τα property-based GUI εργαλεία και τα εργαλεία που χρησιμοποιούν την αναγνώριση εικόνας. Στην πρώτη κατηγορία ανήκουν εργαλεία όπως το Watir, Watin, Cypress και το Selenium, ενώ στη δεύτερη εργαλεία όπως το Sikuli, Protractor, EggPlant, Worksoft κ.α. Υπάρχει και μία τρίτη κατηγορία εργαλείων που χρησιμοποιούν μόνο συντεταγμένες για να αλληλοεπιδράσουν με τις διάφορες εφαρμογές ιστού, και δεν θα αναφερθούμε εκτεταμένα σε αυτά, καθώς πρόκειται για τεχνολογίες που είχαν κάποια χρήση στο παρελθόν, αλλά πλέον εντοπίζονται ως παρωχημένα.

Στην παρούσα διπλωματική εργασία χρησιμοποιήθηκε το Selenium σε μία διερευνητική χρήση με την προσθήκη της βιβλιοθήκης αναγνώρισης εικόνων OpenCVSharp. Η επιλογή αυτή συνάδει με τις δυνατότητες αλλά και την ευρεία αναγνωσιμότητα του Selenium, καθώς βρίσκεται ανάμεσα στις πρώτες θέσεις επιλογής, σε σχέση με πολλά άλλα εργαλεία πάνω από μία δεκαετία. Η δυνατότητά του να αλληλοεπιδρά με τις εφαρμογές ιστού, αλλά και ο χρόνος που το Selenium βρίσκεται ανάμεσα στις πρώτες επιλογές εταιρειών και οργανισμών, δείχνει ότι δεν πρόκειται για ένα νεοεισερχόμενο εργαλείο στο χώρο, όπου σε μερικά χρόνια θα εκλείψει η χρηστικότητα και η συντηρησιμότητα του. Ακόμη, με την πάροδο του χρόνου έχουν λυθεί τα αρχικά προβλήματα που εμφανίζονται σε νέες τεχνολογίες, ενώ υπάρχει μία βεβαιότητα ότι σε νέες εκδόσεις του εργαλείου, όπου είτε έχουν εισαχθεί καινούργιες δυνατότητες ή έχουν λυθεί προβλήματα που ανακύπτουν λόγω της εξέλιξης των τεχνολογιών, δεν θα ανακύψουν θέματα ασυμβατότητας. Θέματα όπου οι μηχανικοί θα χρειάζονται να δαπανήσουν περισσότερες ώρες για τη συντήρηση

του εκάστοτε έργου δοκιμών, ενώ αντίστοιχα ο χρόνος αυτός μπορεί να αφιερωθεί στην ανάπτυξη περισσότερων αυτοματοποιημένων δοκιμών.

Με την ένταξη της βιβλιοθήκης ανοικτού λογισμικού OpenCVSharp, θα προσπαθήσουμε να δημιουργήσουμε μια βιβλιοθήκη, η οποία θα παρέχει το μηχανισμό αναγνώρισης και αλληλεπίδρασης με τα κρυμμένα αντικείμενα (Hidden Elements) του μοντέλου αντικειμένων εγγράφου (Document Object Model), σε μία εφαρμογή ιστού, που υλοποιήσαμε οι ίδιοι για τον σκοπό της εργασίας.

Παρότι η γλώσσα προγραμματισμού Java διατηρεί ακόμα την πρωτοκαθεδρία της, όσον αφορά τη δημιουργία έργων δοκιμών, αλλά και η Python κατέχει πολύ υψηλή θέση, όσο αφορά έργα που συσχετίζονται με την αναγνώριση εικόνων, η εφαρμογή ιστού για τον σκοπό της παρούσας εργασίας υλοποιήθηκε με χρήση της γλώσσας προγραμματισμού C#. Παρατηρήσαμε, τα τελευταία τρία με πέντε χρόνια, τη ραγδαία ανάπτυξη της γλώσσας C#, και τις εκδόσεις .NET core και μετέπειτα .NET5 και άλλες στη συνέχεια, με τις οποίες πλέον υποστηρίζει cross-platform εφαρμογές. Ακόμα, έχει αυξηθεί η χρήση της από μηχανικούς δοκιμών, αφού είναι εξίσου αποτελεσματική στην εκτέλεση και ανάπτυξη δοκιμαστικών σεναρίων.

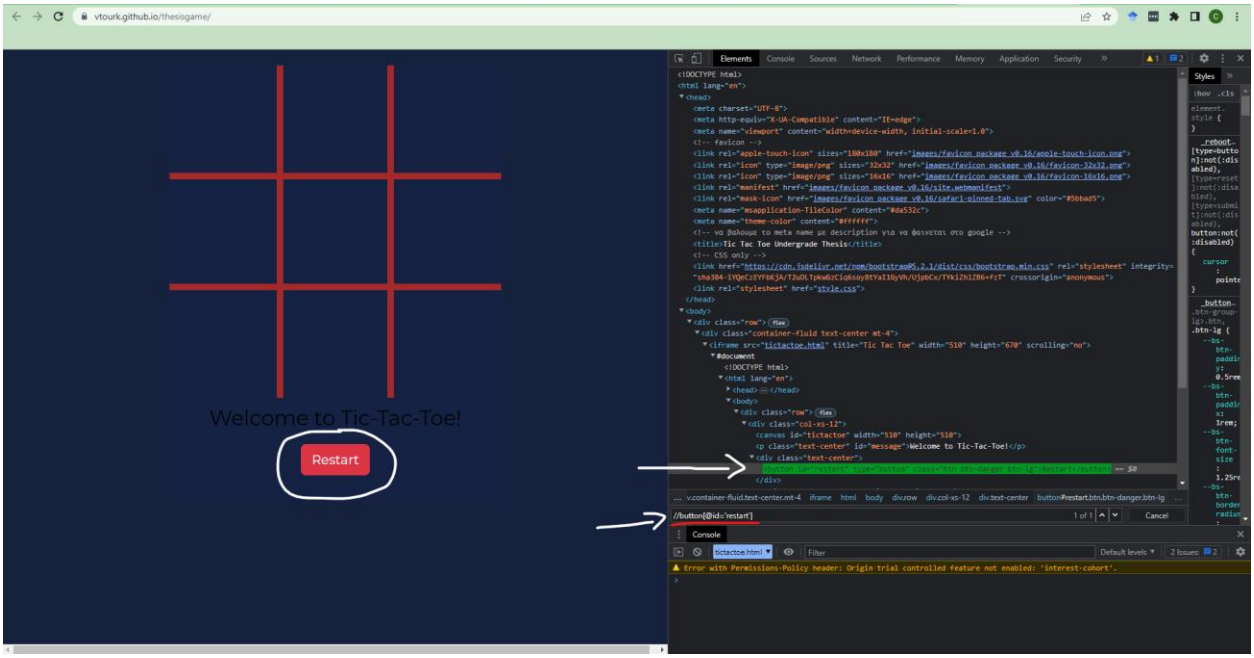
Προχωρήσαμε στην συνέχεια σε μία έρευνα για το ποιες είναι οι διαθέσιμες βιβλιοθήκες, για την αναγνώριση και την επεξεργασία εικόνων, βρίσκοντας την βιβλιοθήκη OpenCVSharp αρκετά ελκυστική, σε σχέση με άλλες επιλογές, όπως η AForge που είναι για μία βιβλιοθήκη wrapper της OpenCV - μία από τις πιο χρησιμοποιούμενες βιβλιοθήκες αναγνώρισης εικόνας, με τη χρήση της Python. Θεωρήσαμε ότι η συγκεκριμένη βιβλιοθήκη με τις ομοιότητες που διαθέτει, μας δίνει τη δυνατότητα για έρευνα και μελέτη, όχι μόνο μέσω των παραδειγμάτων χρήσης της, που μπορούν να βρεθούν στον ιστό, αλλά και μέσω της εύρεσης επεξηγηματικών παραδειγμάτων της ίδιας της OpenCV. Η επιλογή της C# έγινε ώστε να δοθεί μια εξίσου ικανή λύση με αυτή που θα προέκυπτε από άλλους συνδυασμούς επιλογών (π.χ., η Python που υποστηρίζει το Selenium, και διαθέτει την OpenCV ή η Java με χρήση της BoofCV).

2. Μεθοδολογία

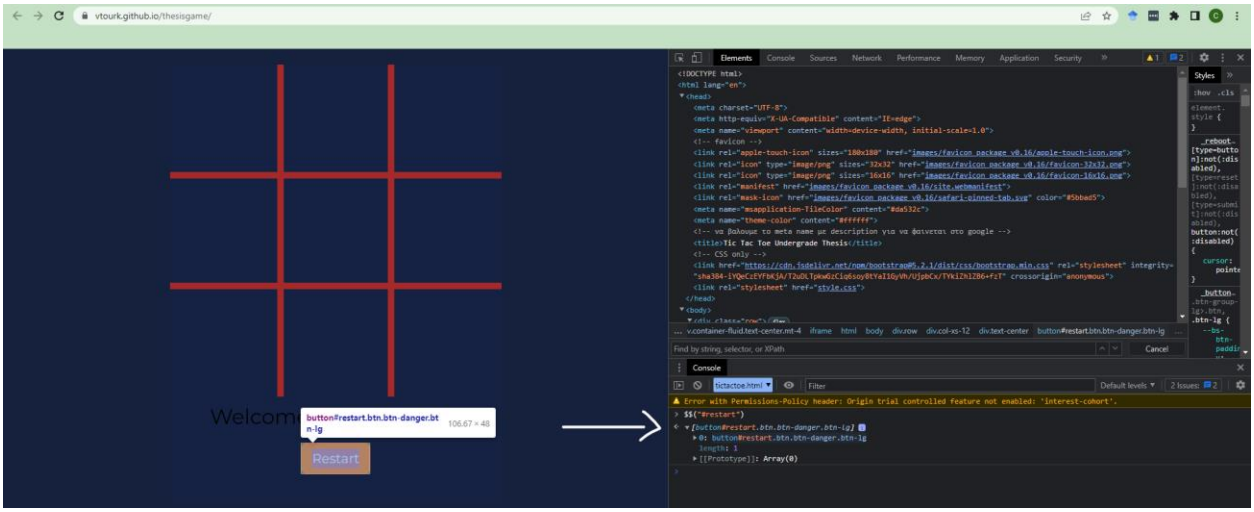
Το Selenium όπως και άλλα εργαλεία δοκιμών προσπαθούν να προσομοιώσουν την αλληλεπίδραση του χρήστη με την εφαρμογή ιστού, όμως στην πραγματικότητα αλληλοεπιδρούν με το σύστημα που βρίσκεται υπό δοκιμή. Συγκεκριμένα το Selenium χρησιμοποιεί για την εύρεση των αντικειμένων δύο ειδών μηχανισμούς, τα διάφορα XPaths και CSS χαρακτηριστικά, που απαρτίζουν την σελίδα ιστού.

Η εύρεση και οι αλληλεπιδράσεις αντικειμένων, που γράφονται σε μία από τις υποστηριζόμενες γλώσσες, γίνονται με HTTP αιτήματα από τον Web Driver. Συνεπώς, υπάρχουν κρυμμένα αντικείμενα, τα ποια δεν μπορούν να εντοπιστούν καθόλου από το Selenium, όπως κρυμμένα αντικείμενα που εμπεριέχονται μέσα σε Canva. Δημιουργείται λοιπόν η αβεβαιότητα του κατά πόσο η απουσία της αναγνώρισης των αντικειμένων, αλλά και ο τρόπος που πραγματοποιούνται οι αλληλεπιδράσεις με τα αντικείμενα του DOM, μπορούν να επηρεάσουν τα αποτελέσματα μίας αυτοματοποιημένης δοκιμής, σε αντίθεση με μία δοκιμή που εκτελείται από έναν άνθρωπο. Αυτό συμβαίνει, διότι ο άνθρωπος είναι ικανός να εντοπίσει εύκολα κάποια σφάλματα, όπου το Selenium δεν θα καταφέρει να τα αναγνωρίσει.

Για το σκοπό της εργασίας, δημιουργήσαμε μία σελίδα ιστού, όπου εμπεριέχει παραδείγματα των δύο τύπων αντικειμένων. Αντικείμενα που δεν είναι κρυμμένα, όπως το κουμπί “RESTART”, που εμπεριέχεται μέσα σε ένα IFrame. Όπως φαίνεται και στις παρακάτω δύο εικόνες, μπορεί να επιτευχθεί η αναγνώριση του αντικειμένου στο DOM.



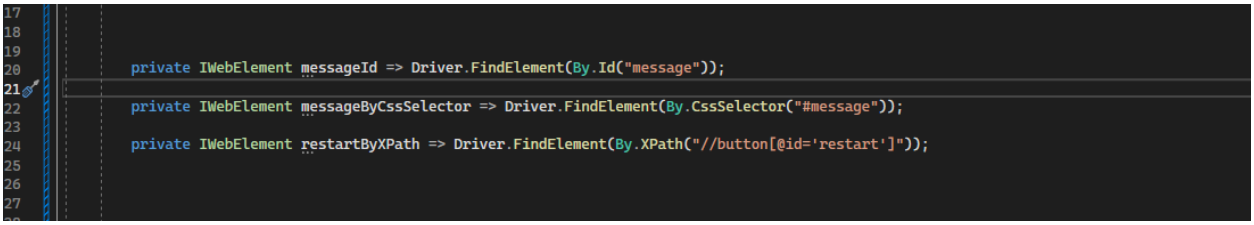
Εικόνα 1: Με χρήση της τοποθεσίας (XPath) του κουμπιού (//button[@id='restart']) στην HTML σελίδα.



Εικόνα 2: Με χρήση των χαρακτηριστικών CSS του κουμπιού (#restart).

Η χρήση του εντοπισμού αντικειμένων με το Selenium framework, καθιστάται δυνατή από την κλήση της public μεθόδου “FindElement”, ενός αντικειμένου τύπου “IWebDriver”. Για να επιτευχθεί η κλήση της μεθόδου, χρειάζεται να περάσουμε παραμετρικά ένα αντικείμενο τύπου “By”. Η κλάση “By“, μας παρέχει τις διάφορες στατικές μεθόδους, από τις οποίες μπορούμε να επιλέξουμε το μηχανισμό που επιθυμούμε να χρησιμοποιήσουμε. Πέρα από τη χρήση των

βασικών μηχανισμών, “By.CssSelector” και “By.XPath”, παρέχονται επίσης οι “By.Id”, “By.ClassName”, “By.Name” και άλλοι. Όλες οι στατικές μέθοδοι, ζητούν να περαστεί παραμετρικά μία μεταβλητή τύπου string. Κατά τη χρήση των “By.CssSelector” και “By.XPath” μεθόδων, το Selenium χρησιμοποιεί αυτούσιο το locator που παρέχουμε για τον εντοπισμό. Στις υπόλοιπες μεθόδους, όπως για παράδειγμα στην “By.Id” και “By.ClassName”, το Selenium προσθέτει μπροστά από το locator τα σύμβολα “#” και “.” αντίστοιχα. Στη συνέχεια προχωράει στον εντοπισμό του αντικειμένου με τη χρήση της CSS.
Στην Εικόνα 3, φαίνονται παραδείγματα της κλήσης των μεθόδων

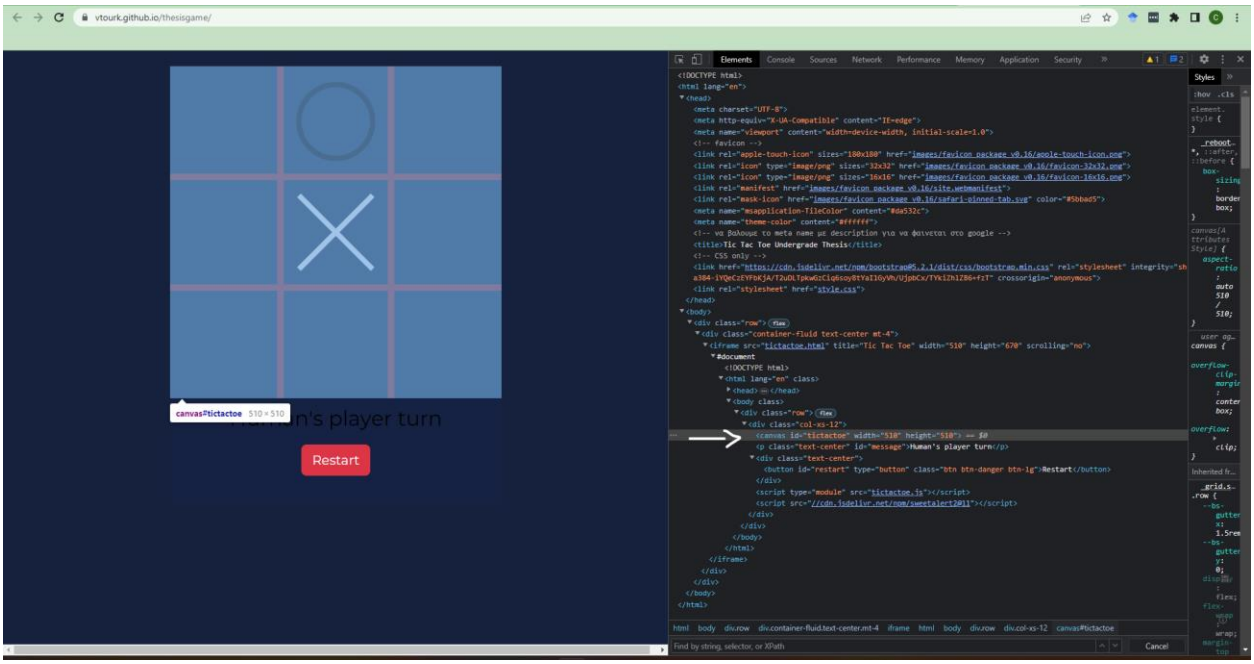


Εικόνα 3: Παραδείγματα κλήσης μεθόδων.

Ο δεύτερος τύπος αντικειμένων που εμπεριέχει η σελίδα, είναι τα κρυμμένα αντικείμενα.

Για να γίνει ρεαλιστικό το παράδειγμα της αναγνώρισης εικόνων, χρειάστηκε να αναπτύξουμε το παιχνίδι της τρίλιζας με την τεχνολογία Canva. Ως αποτέλεσμα της επίτευξης ενός ολοκληρωμένου γύρου του παιχνιδιού, ο παίκτης “X” καλείται να επιλέξει ένα κουτί της αρεσκείας του. Έχοντας χρησιμοποιήσει την τεχνολογία Canva, ο εντοπισμός των κουτιών που εμπεριέχουν ήδη κάποιο σύμβολο “X” και “O” ή ο εντοπισμός ενός άδειου κουτιού, καθιστάται δυνατός για έναν άνθρωπο, αλλά δεν μπορεί να επιτευχθεί με κάποιον από τους προσφερόμενους μηχανισμούς του Selenium.

Στην Εικόνα 4 φαίνεται η ανυπαρξία των διάφορων αντικειμένων του παιχνιδιού στο DOM.



Εικόνα 4: Ανυπαρξία των αντικειμένων παιχνιδιού στο DOM.

Καλούμαστε λοιπόν, να αναπτύξουμε μία βιβλιοθήκη η όποια θα παρέχει έναν μηχανισμό εντοπισμού των τετραγώνων, όπως ακόμα και κάποιες μεθόδους αλληλεπίδρασης με τα κρυμμένα στοιχεία της εφαρμογής. Με το μηχανισμό εντοπισμού θα μπορούμε να διακρίνουμε ποια τετράγωνα θα μπορούν να επιλεγούν κατά τη δοκιμή, ενώ ο μηχανισμός αλληλεπίδρασης θα επιτρέψει στη δοκιμή να επιλέγει ένα άδειο ή ένα γεμάτο τετράγωνο, αναλόγως το εκάστοτε εκτελούμενο σενάριο.

3. Τεχνολογίες που Χρησιμοποιήθηκαν

3.1 Περιβάλλον Δοκιμών

Για να επιτευχθεί η αυτοματοποίηση εφαρμογών ιστού, χρειάζεται να γίνει η ανάπτυξη μιας web εφαρμογής που στην προκειμένη περίπτωση θα είναι μία τρίλιζα. Σε αυτή την τρίλιζα θα μπορεί να παίξει ένας άνθρωπος εναντίον υπολογιστή. Αυτό το παιχνίδι θα είναι φτιαγμένο με τέτοιο τρόπο ώστε κάποια elements να είναι hidden (μέσα σε iframe) όπου θα γίνεται η αναγνώριση των tiles με χρήση εικόνων, ενώ θα γίνεται και η χρήση των locators HTML / CSS / XPATH σε κομμάτια όπως το κουμπί και το pop up message.

3.2 Εισαγωγή στο Front-End Development

Το Front-end σε ένα website είναι το κομμάτι που ένας χρήστης μπορεί να έχει αλληλεπίδραση. Οτιδήποτε μπορεί να δει ο χρήστης κατά τη διάρκεια της περιήγησης, δηλαδή από fonts, χρώματα, dropdown menus και sliders γίνεται με τον συνδυασμό των Front-end εργαλείων και γλωσσών προγραμματισμού, όπως HTML, CSS και Javascript που θα αναλυθούν παρακάτω.

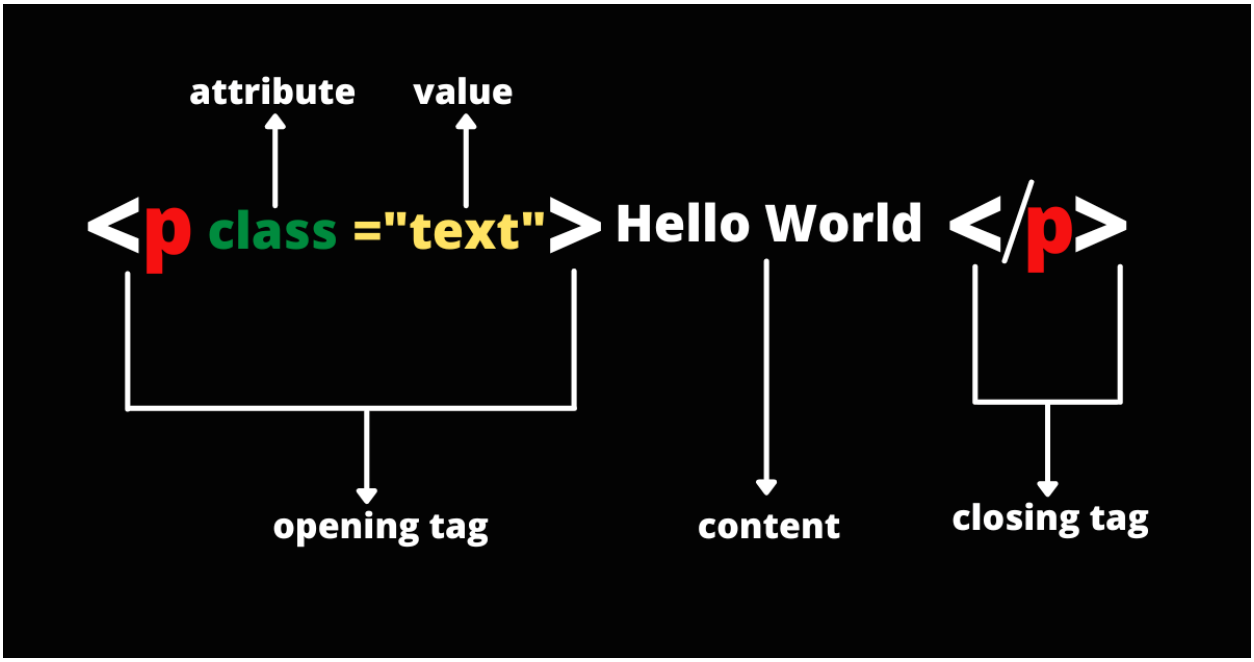
3.3 Ορισμός για την HTML

Η HTML (Hyper Text Markup Language, ή αλλιώς Γλώσσα Σήμανσης Υπερκειμένου) είναι η κύρια γλώσσα που χρησιμοποιείται για ιστοσελίδες. Σκοπός της HTML είναι ο Web browser να διαβάζει τα έγγραφα HTML (που έχουν κατάληξη σε .html) ώστε να σχηματιστεί η ιστοσελίδα σε μορφή που οποιοσδήποτε μπορεί να διαβάσει. Η πιο διαδεδομένη version της HTML είναι η HTML5 καθώς σε σχέση με άλλες εκδόσεις, έχει λύσει θέματα συμβατότητας που υπήρχαν παλαιότερα.

Δομικά η HTML αποτελείται από ετικέτες ή αλλιώς Tags, τα οποία είναι ενθυλακωμένα μέσα στα σύμβολα ισότητας και ανισότητας ‘<‘ ’>’ όπου αυτό επιτυγχάνει τον web browser να εμφανίσει σωστά το περιεχόμενό του. Αυτές οι ετικέτες είναι συνήθως σε ζεύγη (πχ <h1> **αυτό είναι ένα header (επικεφαλίδα)** </h1>) όπου η πρώτη ονομάζεται ετικέτα έναρξης και η δεύτερη λήξης.

Ανάμεσα στο ζεύγος, ένας developer μπορεί να ξεκινήσει να σχεδιάζει την ιστοσελίδα του, τοποθετώντας πίνακες, κείμενα, εικόνες κ.α.

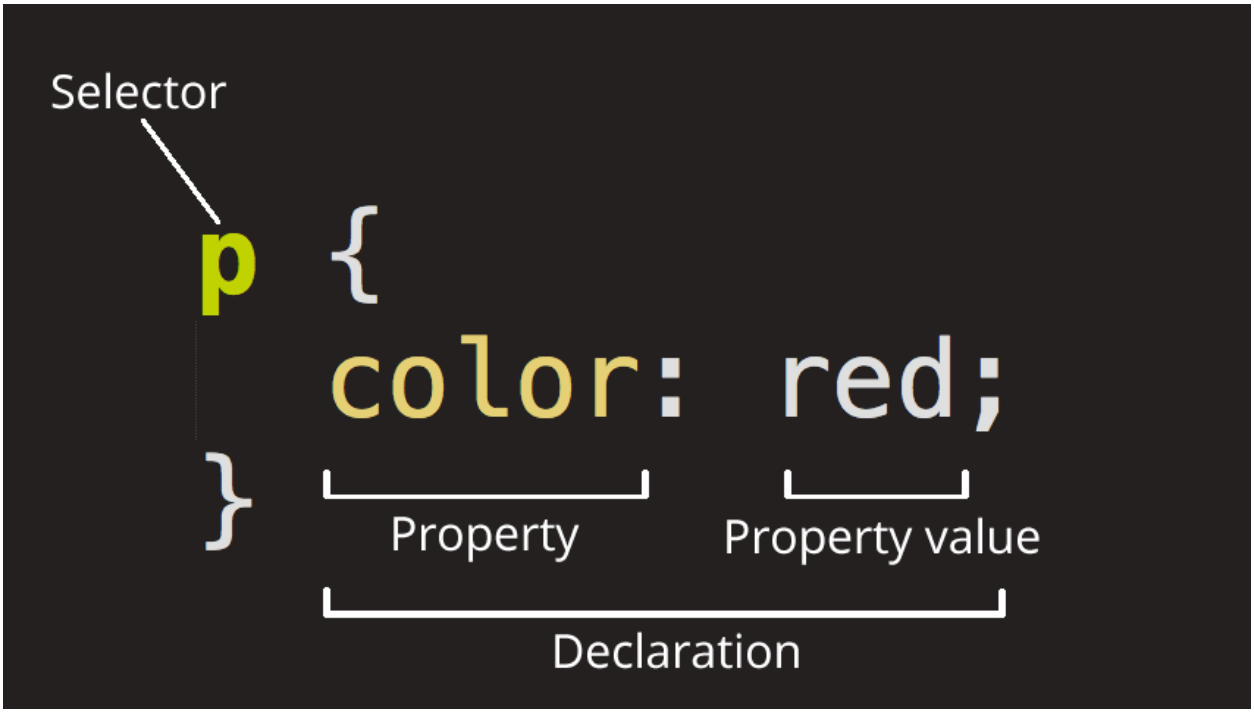
Ο κώδικας ξεκινάει με το <!DOCTYPE html> το οποίο είναι μία πληροφορία για τον browser ώστε να ξέρει τι να περιμένει, δηλαδή το πως είναι γραμμένη η ιστοσελίδα. Τα έγγραφα HTML ξεκινούν και ολοκληρώνονται με τις ετικέτες <html> και </html> αντίστοιχα που δηλώνει ότι ο κώδικας που ακολουθεί είναι της HTML. Μέσα του θα βρούμε τα tags head & body. Το <head> </head> tag είναι ένα «δοχείο» για τις βασικές πληροφορίες της ιστοσελίδας όπως, στοιχεία του τίτλου, style, link, scripts κ.α. Στην συνέχεια το tag <body> </body> εμπεριέχει το κύριο μέρος του κώδικά μας. Το body αποτελείται από elements όπως οι επικεφαλίδες <h1> </h1>, παραγράφους , τμήματα και άλλα.



Εικόνα 5: HTML.

3.4 Ορισμός για την CSS

CSS (Cascading Style Sheets ή αλλιώς διαδοχικά φύλλα ύφους) είναι μια γλώσσα υπολογιστή που χρησιμοποιείται για να δώσει ύφος σε ένα HTML αρχείο, δηλαδή περιγράφει πως ένα αρχείο HTML θα πρέπει να εμφανίζεται σε μια ιστοσελίδα. Η αποθήκευση των αρχείων γίνεται σε ξεχωριστό αρχείο υπό την μορφή .css και καλείται με την χρήση ενός <link></link> από στην HTML. Η CSS βασίζεται σε 2 βασικούς κανόνες. Ένας κανόνας της αποτελείται από τον selector (επιλογέα) ο οποίος μπορεί να είναι μία ετικέτα HTML την οποία θέλουμε να διαμορφώσουμε. Μέσα του κάθε selector περιέχει τις CSS properties (ιδιότητες) οι οποίες αφορούν για το πως θα μορφοποιηθεί αυτός ο selector.



Εικόνα 6: CSS.

3.5 Ορισμός για την Bootstrap

Η Bootstrap είναι ένα open-source CSS framework, το οποίο είναι responsive, δηλαδή μπορεί κάποιος να το χρησιμοποιήσει και να έχει ομοιόμορφη απεικόνιση της ιστοσελίδας σε διαφορετικές οθόνες χωρίς να αλλοιώνεται το content και η μορφή. Εμπεριέχει HTML, CSS και Javascript-based σχεδιαστικά template, για buttons, forms, navigation, typography αλλά και άλλα interface components.

Αν κάποιος θέλει να συμπεριλάβει την Bootstrap στο build του, υπάρχουν 2 τρόποι για να γίνει αυτό. Είτε την κατεβάζει locally, είτε μπορεί να την συμπεριλάβει με το CDN link το οποίο δεν απαιτείται κάποιου τύπου εγκατάσταση. Το CDN πρακτικά είναι content delivery network, το οποίο αναφέρεται σε ένα γκρουπ από servers οι οποίοι δουλεύουν μαζί για να παρέχουν γρήγορη παράδοση του περιεχομένου.

Η Bootstrap έχει γίνει developed για τις κινητές συσκευές (μία στρατηγική για να είναι πιο optimal ο κώδικας) και στη συνέχεια κάνει scale up για μεγαλύτερες οθόνες χρησιμοποιώντας αναγκαία CSS media queries.

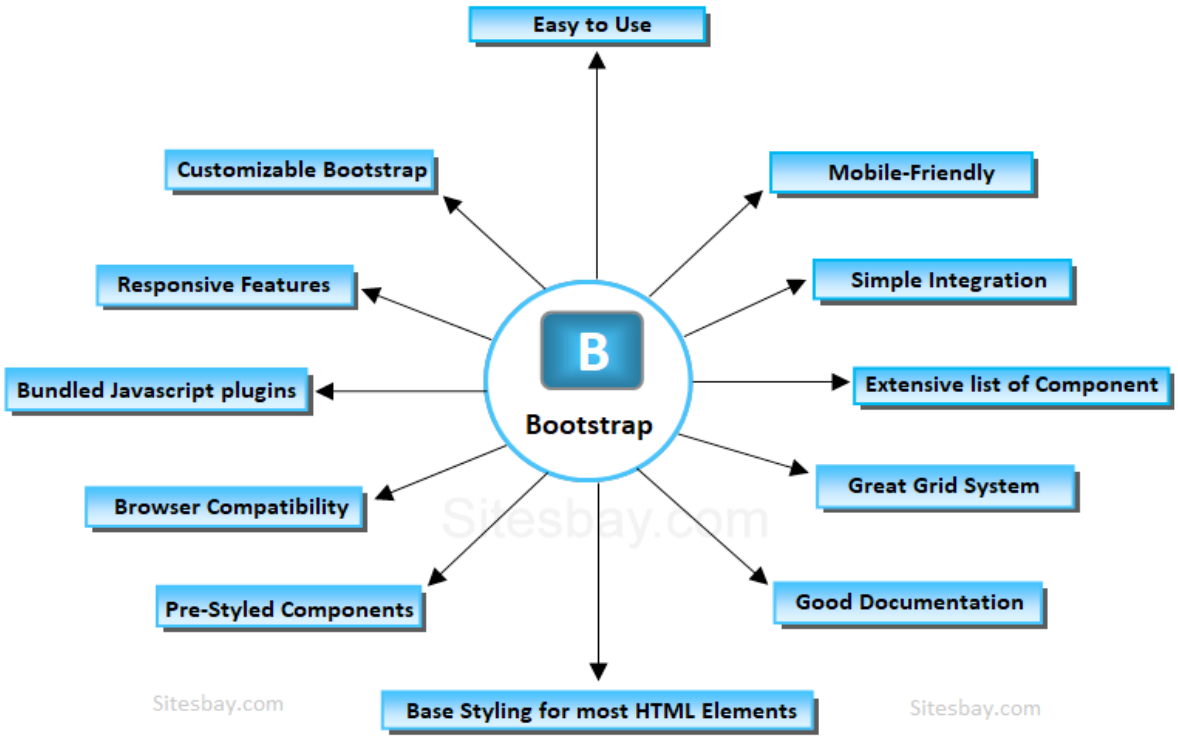
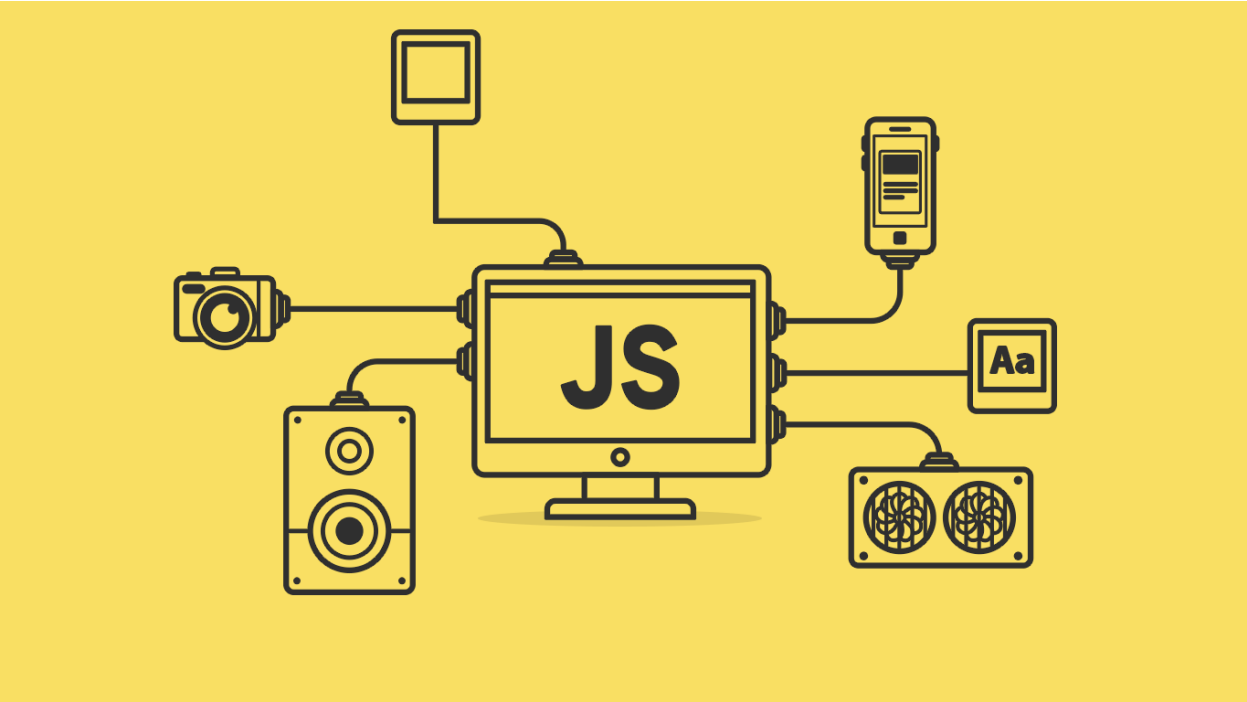


Figure 7: Διάγραμμα Λειτουργιών Bootstrap.

3.6 Ορισμός για την JavaScript

Η Javascript είναι μια διερμηνευμένη γλώσσα προγραμματισμού. Είναι γλώσσα σεναρίων (scripting language) που βασίζεται στα πρωτότυπα (με διαφορετικό ύφος αντικειμενοστρέφειας καθώς επαναχρησιμοποιεί ήδη υπάρχων αντικείμενα). Επίσης είναι η πιο διαδεδομένη γλώσσα που χρησιμοποιείται παγκοσμίως στο Web. Σε αντίθεση με την HTML όπου καθορίζει το περιεχόμενο μιας ιστοσελίδας, η Javascript χρησιμοποιείται για να προγραμματίσει την συμπεριφορά των ιστοσελίδων.

Το βασικότερο πλεονέκτημα της Javascript σε σχέση με άλλες γλώσσες είναι ότι δεν χρειάζεται compiler intepreted αλλά μόνο η παρουσία ενός απλού browser για να εκτελεστεί.



Εικόνα 8: Εφαρμογές Java.

3.7 Δομή της JavaScript

Ο κώδικας της Javascript πρέπει να βρίσκεται μέσα στον κώδικα της HTML, στον οποίο παρέχει δυνατότητες που η HTML δεν έχει. Αυτό επιτυγχάνεται με την χρήση του tag < script> όπου μέσα σε αυτό θα βάζουμε τις εντολές της. Συνήθως όμως αυτό που γίνεται είναι η χρήση ενός ‘src’ όπου

είναι ένα χαρακτηριστικό που προσδιορίζει το URL από ένα εξωτερικό Javascript αρχείο. Αυτό επίσης είναι χρήσιμο αν θες να καλέσεις το ίδιο Javascript αρχείο πολλές φορές σε διαφορετικές σελίδες.

3.8 Canvas

Στην συγκεκριμένη πτυχιακή εργασία, χρησιμοποιείται ο canvas όπου είναι ένα element που για την απεικόνιση 2D γραφικών μέσω μιας scripting γλώσσας (συνήθως Javascript), Παραδείγματος χάρη, να ζωγραφιστούν γραφικά, ένωση φωτογραφιών ή παραγωγή απλών animations.

3.9 Ορισμός του Node JS

Το node.js, είναι μία ανοιχτού λογισμικού πλατφόρμα ανάπτυξης λογισμικού για το διαδίκτυο σε περιβάλλον Javascript. Μας επιτρέπει να εκτελέσουμε Javascript κώδικα έξω από τον Web browser (φυλλομετρητή), όπου συνήθως τρέχει η Javascript.
Η Node.js έχει τη δυνατότητα δημιουργίας εργαλείων και βιβλιοθηκών όπου τρέχουν στους διακομιστές (servers) , ώστε να μπορεί να δημιουργηθεί δυναμικά το περιεχόμενο της ιστοσελίδας πριν σταλεί στον Web browser του χρήστη. Αυτό που επί της ουσίας γίνεται είναι η ανάπτυξη ενός λογισμικού, το οποίο στην συνέχεια θα συμπυκνωθεί και θα κάνει την ανάπτυξη της ιστοσελίδας αλλά και των διαδικτυακών εφαρμογών, να γίνεται με μόνο μια γλώσσα, την Js.
Επίσης έχει event-driven, non-blocking I/O μοντέλο, δηλαδή η ροή προγράμματος εξαρτάται από εισερχόμενα γεγονότα, ασύγχρονη είσοδο και έξοδο, όπου δεν χρειάζεται να εκτελεστεί το κομμάτι του κώδικα γραμμή προς γραμμή. Αυτός ο συνδυασμός αυτού του μοντέλου το κάνει να είναι ελαφρύ, γρήγορο και αποδοτικό σε πραγματικού χρόνου Web εφαρμογές.
Οι εφαρμογές του node.js χαρακτηρίζονται ως state-of-the-art.

3.10 NPM Package Manager

Το NPM ή αλλιώς Node Package Manager είναι ο διαχειριστής των πακέτων για την Javascript, η προεπιλογή του Node.js και είναι το μεγαλύτερο Software Registry. Είναι ανοιχτού κώδικα και χρησιμοποιείται από προγραμματιστές από όλο τον κόσμο για να μοιραστούν ή να δανειστούν

packages (βιβλιοθήκες), αλλά και από οργανισμούς για την διαχείριση των ιδιωτικών τους packages.

Το NPM διακρίνεται στις εξής 3 κατηγορίες:

- Το *website* όπου βρίσκεις τα packages τα οποία θες, το στήσιμο των profile, αλλά και την διαχείριση παραμέτρων. Πχ, μπορείς να δώσεις πρόσβαση σε οργανισμούς να διαχειρίζονται τις προσβάσεις είτε σε δημόσια, είτε σε ιδιωτικά packages.
- Το *CLI* (Command Line Interface) όπου τρέχει από το Terminal, και είναι το μέρος όπου ο προγραμματιστής επιτυγχάνει την αλληλεπίδραση με το Npm.
- Το *Registry* που είναι μια μεγάλη βάση δεδομένων από Javascript λογισμικό και το meta-information (σχόλια) που περιβάλλουν το λογισμικό.

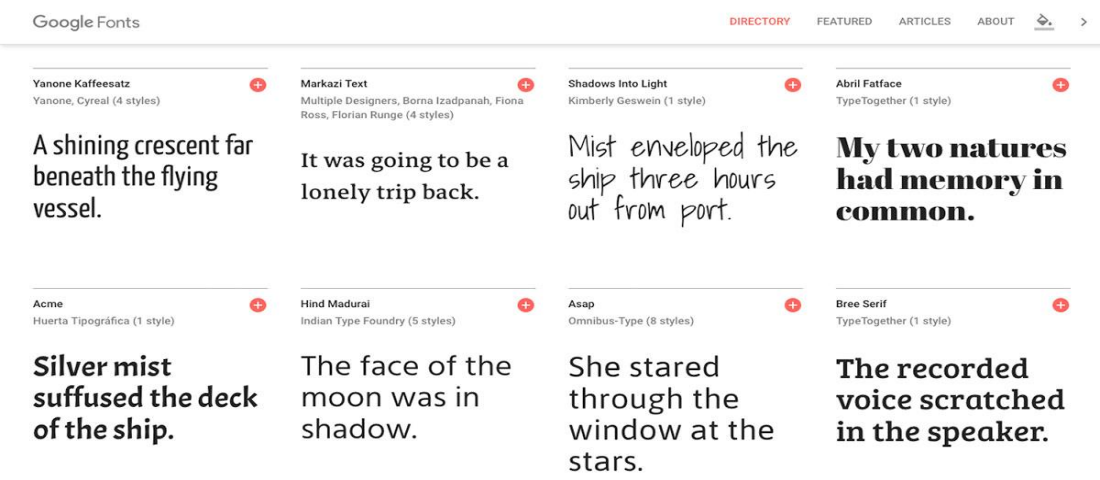
3.11 Package JSON

Το package Json, είναι ένα JSON αρχείο το οποίο βρίσκεται στο root του Javascript/Node project. Αυτό το αρχείο εμπεριέχει τα metadata που είναι σχετικά με το project και χρησιμοποιούνται για την διαχείριση των dependencies, scripts, version αλλά και διαφόρων άλλων παραμέτρων.

Υπάρχουν 2 τρόποι να δημιουργηθεί ένα package Json αρχείο, με το Npm το οποίο έχει χρησιμοποιηθεί σε αυτή την πτυχιακή εργασία ή με το Yarn.

3.12 Google Fonts

Τα Google fonts, είναι ένας εύκολος τρόπος να αλλάξεις γραμματοσειρές, καθώς είναι open source και προσφέρονται δωρεάν. Σου δίνουν την δυνατότητα να επιλέξεις ανάμεσα σε παραπάνω από 135 γλώσσες, αλλά και να τις προσαρμόσεις στις ανάγκες σου. Όταν επιλέξεις τη γραμματοσειρά που θες, τοποθετείς το link που θες στο head της HTML, ή μπορείς να κάνεις import απευθείας στο style της CSS. Στη συνέχεια, μέσα στην CSS, κάνεις specify το style το οποίο θες.



Εικόνα 9: Google Fonts.

3.13 GitHub Pages

Το GitHub Pages είναι ένα στατικό website, εξυπηρετητής που δέχεται HTML, CSS και Javascript αρχεία απευθείας από το repository (αποθήκη) του GitHub. Στην συνέχεια τρέχει τα αρχεία από ένα build process και δημοσιεύει το website. Σκοπός του GitHub Pages είναι να παρέχει στους χρήστες του, έναν τρόπο να δημιουργήσουν μια προσωπική ιστοσελίδα για τους ίδιους και τα project τους. Επιπλέον αυτή η υπηρεσία παρέχεται δωρεάν.

3.14 Iframes

Τα iframes (inline frames) είναι ένα HTML στοιχείο δεδομένων, όπου φορτώνει και ενσωματώνει μέσα σε ένα καθορισμένο πλαίσιο μία άλλη HTML σελίδα. Χρησιμοποιούνται συχνά για ενσωματωμένα βίντεο, διαδραστικούς πίνακες διαφημίσεις κ.α.

4. Εργαλείο Αυτοματισμού Δοκιμών

4.1 Η Γλώσσα Προγραμματισμού C#

Η C# (“See Sharp”) είναι μία μοντέρνα αντικειμενοστραφής γλώσσα προγραμματισμού που προέρχεται από τη γλώσσα προγραμματισμού C. Έχει δημιουργηθεί και συντηρείται από την Microsoft, ενώ ανήκει σε μία ευρύτερη οικογένεια γλωσσών μαζί με τις C, C++, Java κ.α. Φαίνεται ότι υπάρχει μεγάλη ομοιότητα μεταξύ της C# και των γλωσσών C++ και Java, καθώς η πρώτη έχει δανειστεί ή ακόμα και εξελίξει αρκετά από χαρακτηριστικά τους.

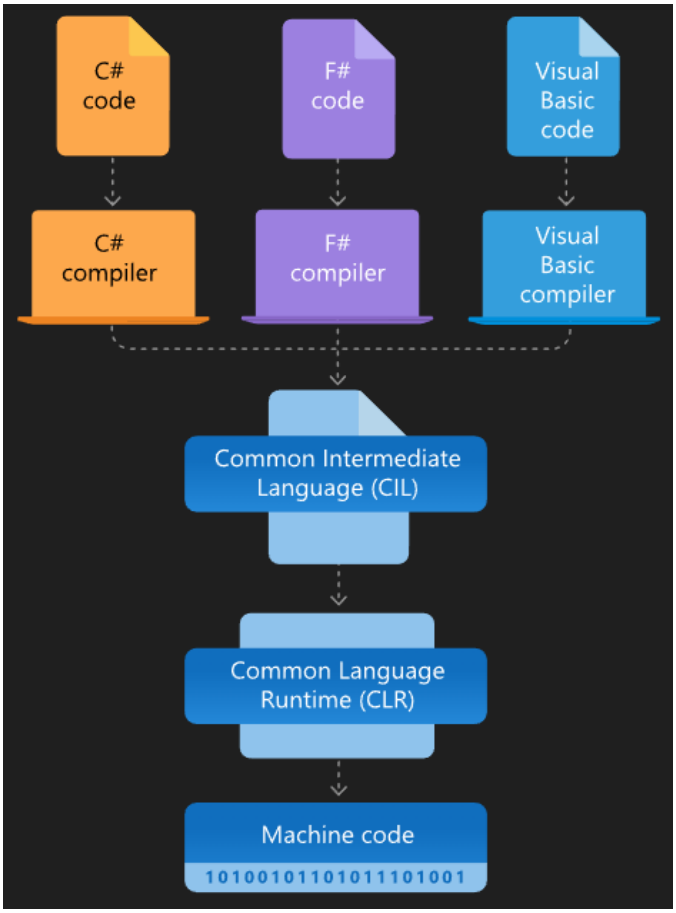
Είναι μία “compiled” γλώσσα προγραμματισμού. Αυτό σημαίνει ότι για να μπορέσει να τρέξει σε έναν υπολογιστή επιβάλλεται πρώτα να περάσει από ένα layer μετάφρασης, όπου ένα πρόγραμμα (compiler) θα την μετατρέψει σε μία πιο χαμηλού επιπέδου γλώσσα μηχανής. Πρόκειται για μία σύγχρονη strongly typed γλώσσα όπου όλα τα αντικείμενα και οι κύριοι τύποι όπως int, double, string κ.α. κληρονομούν από μόνο ένα αντικείμενο, ενώ υποστηρίζει τον ασύγχρονο προγραμματισμό εφαρμογών. Συνεχίζει να εμπλουτίζεται διαρκώς με αποτέλεσμα να μπορεί ο προγραμματιστής να εφαρμόζει ακόμα και τα πιο σύγχρονα σχεδιαστικά πρότυπα μέσα στις εφαρμογές που αναπτύσσει.

Εικόνα 10: Εφαρμογές C#.

4.2 Η Πλατφόρμα .NET

Η .NET (“Dot Net”) είναι μία από τις πιο γνωστές πλατφόρμες που παρέχονται δωρεάν και είναι ανοικτού κώδικα. Δημιουργήθηκε από την Microsoft και συντηρείται από την ίδια, το .Net Foundation και την .Net κοινότητα ανοιχτού κώδικα. Υποστηρίζει την ανάπτυξη πολλών διαφορετικών τύπων εφαρμογών όπως για παράδειγμα είναι οι mobile, desktop, web και console εφαρμογές. Οι προαναφερθείσες φιλοξενούνται σε συστήματα Windows αλλά μπορούν να είναι και cross platform εφαρμογές που μπορούν να φιλοξενηθούν σε συστήματα Linux, iOS και Android. Επίσης, τα τελευταία χρόνια υποστηρίζει την Unity, ένα 2D και 3D εργαλείο που έχει εφαρμογή στην ανάπτυξη βιντεοπαιχνιδιών, first-responder εφαρμογές, προσομοιωτές εκπαίδευσης και γενικότερα εφαρμογές αλληλεπίδρασης με 2D και 3D διαστάσεις.

Οι υποστηριζόμενες γλώσσες από την πλατφόρμα είναι η C#, F# και η Visual Basic. Οι γλώσσες μεταφράζονται στην Common Intermediate Language (CIL), μία byte κώδικα γλώσσα και στη συνέχεια, το Common Language Runtime (CLR) αναλαμβάνει την μετάφραση σε κώδικα μηχανής. Η πλατφόρμα παρέχει μία ποικιλία από βιβλιοθήκες, οι οποίες με την σειρά τους παρέχουν έτοιμες λύσεις και συγκροτούν ένα χρήσιμο σύνολο από επιλογές έτοιμες προς χρήση από τον εκάστοτε μηχανικό. Σε κάθε σύγχρονη πλατφόρμα προγραμματισμού, σημαντικό ρόλο παίζει και ένα σύστημα μέσω του οποίου οι προγραμματιστές μπορούν να χρησιμοποιήσουν ή να μοιραστούν πακέτα τα οποία περιέχουν compiled κώδικα, DLLs (Dynamic Link Library). Για αυτό αξίζει να αναφέρουμε ότι για την .NET το υποστηριζόμενο σύστημα διαμοίρασης αρχείων είναι το NuGET, όπου βρίσκεται ενσωματωμένο σε IDE προγράμματα, όπως το Visual Studio 2022 αλλά μπορεί να χρησιμοποιηθεί και μέσω ενός CLI (Command Line Interface).Το σύστημα διαχείρισης πακέτων έχει δημιουργηθεί και υποστηρίζεται από την Microsoft.



Εικόνα 11: Διάγραμμα λειτουργίας πλατφόρμας .NET.

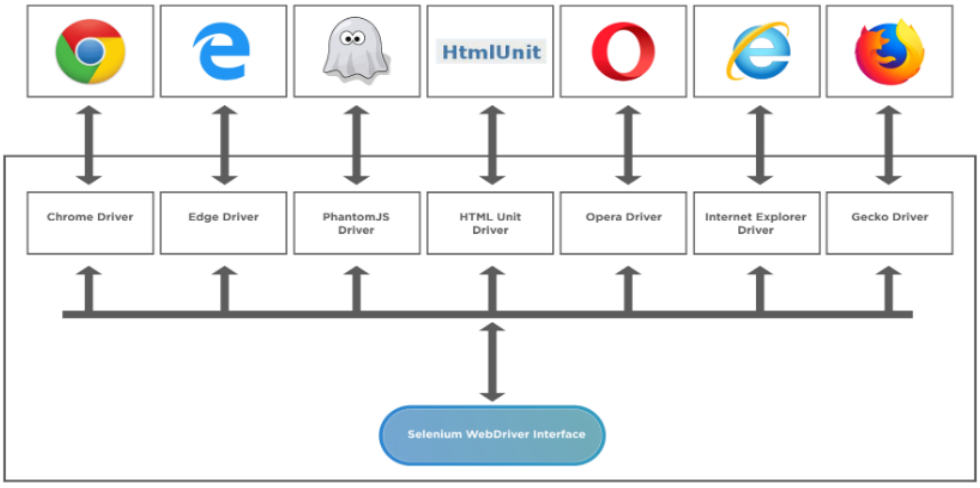
4.3 Εισαγωγή στο Selenium

Το Selenium είναι ένα ανοικτού κώδικα εργαλείο δοκιμών, όπου παρέχει ένα εύρος βιβλιοθηκών, τα οποία στοχεύουν στον αυτοματισμό εφαρμογών ιστού. Αναπτύχθηκε σαν ένα εσωτερικό εργαλείο της εταιρείας ThoughtWorks από τον Jason Huggins και υποστηρίζει διάφορες γλώσσες προγραμματισμού όπως η Java, Python, JavaScript, Ruby, C# κ.α.

Στις πρώτες εκδόσεις του, για να επιτευχθεί η αλληλεπίδραση με μία εφαρμογή ιστού, υπήρχε στο ενδιάμεσο ένα στρώμα επικοινωνίας με έναν εξυπηρετητή το Selenium RC (Remote Control). Κατά την διάρκεια του χρόνου υπήρξε μία ανακατασκευή (refactor) της παλαιάς αρχιτεκτονικής του εργαλείου με μία νέα, καθιστώντας μη αναγκαία την χρήση ενός εξυπηρετητή για την εκτέλεση δοκιμών. Πλέον βασικό συστατικό για να καταστεί δυνατή η επικοινωνία με το πρόγραμμα περιήγησης (client), είναι η ύπαρξη του Selenium WebDriver. Πρόκειται για μία

διεπαφή (interface) που εμπεριέχει διάφορες εντολές. Οι εντολές αυτές μεταβιβάζονται με την σειρά τους σε ένα οδηγό περιήγησης (browser driver). Με αυτήν την αρχιτεκτονική έχει επιτευχθεί η υποστήριξη των πιο γνωστών προγραμμάτων περιήγησης όπως το Google Chrome, Edge με την χρήση του οδηγού περιήγησης ChromeDriver, του Firefox με την χρήση του GeckoDriver και άλλων.

Προσφέρει ακόμα τη δυνατότητα στους μηχανικούς μέσω της χρήσης του Selenium Grid, για εκτέλεση των δοκιμών σε απομακρυσμένα μηχανήματα, χρησιμοποιώντας οποιουσδήποτε πιθανούς συνδυασμούς από εφαρμογές περιήγησης και λειτουργικών συστημάτων θελήσουν να χρησιμοποιήσουν.



Εικόνα 12: Selenium.

4.4 Η Βιβλιοθήκη OpenCVSharp

Η OpenCVSharp (“Open Source Computer Vision Library See Sharp”) για μία wrapper βιβλιοθήκη της OpenCV (“Open Source Computer Vision Library”) καθιστά δυνατή τη χρήση μεθόδων και λειτουργιών της, με τη γλώσσα προγραμματισμού C# και την .NET πλατφόρμα. Η OpenCV είναι μια βιβλιοθήκη λειτουργιών προγραμματισμού που στοχεύουν κυρίως στην όραση υπολογιστή σε πραγματικό χρόνο. Αρχικά αναπτύχθηκε από την Intel, αργότερα υποστηρίχθηκε από την Willow Garage και στη συνέχεια από την Itseez, η οποία στην εξαγοράστηκε από την Intel. Η OpenCV είναι μία cross-platform βιβλιοθήκη που υποστηρίζει γλώσσες όπως C++ και Python. Φιλοξενείται σε δημόσιο αποθετήριο στο GitHub και είναι δωρεάν για χρήση υπό την άδεια ανοιχτού κώδικα Apache 2. Ξεκινώντας από το 2011, το OpenCV παρέχει κάποια

χαρακτηριστικά όπως για παράδειγμα η επιτάχυνση της GPU για λειτουργίες σε πραγματικό χρόνο.

Παρότι η OpenCVSharp είναι μία βιβλιοθήκη wrapper και λιγότερο γνωστή από την OpenCV, υποστηρίζεται για αρκετό καιρό παρέχοντας στους μηχανικούς που την επιλέγουν αρκετές από τις δυνατότητες της OpenCV.

Εικόνα 13: OPENCVSHARP.

5. Ανάπτυξη Web Εφαρμογής

Σε αυτό το κεφάλαιο θα αναλύσουμε την δομή του project καθώς και το τι εμπεριέχει αλλά και κάνει κάθε αρχείο.

5.1 Το Αρχείο tictactoe.js

Το tictactoe.js είναι το αρχείο το οποίο έχει μέσα όλο παιχνίδι μας σε Javascript.
Στην αρχή γίνεται η δήλωση των μεταβλητών καθώς και mapping του παιχνιδιού. Στην εν λόγω πτυχιακή εργασία, τα winning patterns έχουν πάρει τη μορφή μέσα σε έναν πίνακα, όπως αν θα ήταν μάσκες δικτύων. Αν συμπληρωθεί ένας πίνακας με έναν από τους παρακάτω τρόπους, θα προκύψει ο νικητής.
Παρακάτω βλέπουμε ότι γίνεται η χρήση των Event Listeners ώστε να μπορέσουμε να βάλουμε το πάτημα του ποντικιού. Πιο μετά η draw function όπου μέσα της έχει την drawboard, drawX και την drawO. Η function play, function checkWin αλλά και η function winConditions έχουν γίνει για να κάνουν τους απαραίτητους ελέγχους σε κάθε γύρο του παιχνιδιού, π.χ. στην function play, παίρνουμε από την κονσόλα ποιανού παίχτη είναι η σειρά, (ανθρώπου ή υπολογιστή), στην checkWin, γίνεται ο έλεγχος για το αν υπάρχει νικητής αυτό τον γύρο ή αν το παιχνίδι θα είναι ισοπαλία. Τέλος η function popUpMsg είναι ένα πιο όμορφο, responsive, αλλά και customizable feature στα ήδη υπάρχων pop up messages της Javascript.

```
//elements
const canvas = document.getElementById('tictactoe');
const ctx = canvas.getContext('2d');
const msg = document.getElementById('message');
const restartButton = document.getElementById("restart");

const empty =  0, X = 1, O = -1;
const humanPlayer = X;
const computer = O;
const cellSize = 170;
const winPatterns = [
  0b111000000, 0b000111000, 0b000000111, // Horizontally
  0b100100100, 0b010010010, 0b001001001, // Columns
  0b100010001, 0b001010100, // Diagonals like mask
```

```
];

let endGame = false;
var mapBoard = [0,0,0,
                0,0,0,
                0,0,0];

let mouse = {
  x: -1,
  y: -1,};

canvas.width = canvas.height= 3 * cellSize;

draw();
//EventListeners
canvas.addEventListener('mouseout', function () {
  mouse.x = mouse.y = -1;
});

canvas.addEventListener('mousemove', function (e) {
  let x = e.pageX - canvas.offsetLeft,
      y = e.pageY - canvas.offsetTop;

  mouse.x = x;
  mouse.y = y;
});

canvas.addEventListener('click', function (e) {
  play(getCellByCoords(mouse.x, mouse.y));
});

restartButton.addEventListener("click", () =>{
  mapBoard = [
      0, 0, 0,
      0, 0, 0,
      0, 0, 0,
  ],
  endGame = false;
  msg.textContent = "New game, Hunan's Turn";
});

function draw(){
  ctx.clearRect(0, 0, canvas.width, canvas.height); //tha to valei panw deksia, thelei translate
logika
  drawBoard();
```

```
  xando();
  requestAnimationFrame(draw);
}

function drawBoard(){
  ctx.strokeStyle = 'brown';
  ctx.lineWidth= 10;

  ctx.beginPath();
  ctx.moveTo(cellSize, 0);
  ctx.lineTo(cellSize, canvas.height);
  ctx.stroke();

  ctx.beginPath();
  ctx.moveTo(cellSize * 2, 0);
  ctx.lineTo(cellSize * 2, canvas.height);
  ctx.stroke();

  ctx.beginPath();
  ctx.moveTo(0, cellSize);
  ctx.lineTo(canvas.width, cellSize);
  ctx.stroke();

  ctx.beginPath();
  ctx.moveTo(0, cellSize * 2);
  ctx.lineTo(canvas.width, cellSize * 2);
  ctx.stroke();
}

function xando(){
  ctx.lineWidth= 8;

  for (let i = 0; i < mapBoard.length; i++) {
     let coords = getCellCoords(i);
     ctx.save();
     ctx.translate(coords.x + cellSize / 2, coords.y + cellSize / 2);
     if (mapBoard[i] == X) {
        ctx.strokeStyle = 'white';
        drawX();
     } else if (mapBoard[i] == O) {
        ctx.strokeStyle = 'black';
        drawO();
     }

     ctx.restore();
```

```
  }
}

function drawX () {
  ctx.beginPath();
  ctx.moveTo(-cellSize / 3, -cellSize / 3);
  ctx.lineTo(cellSize / 3, cellSize / 3);
  ctx.moveTo(cellSize / 3, -cellSize / 3);
  ctx.lineTo(-cellSize / 3, cellSize / 3);
  ctx.stroke();
}

function drawO () {
  ctx.beginPath();
  ctx.arc(0, 0, cellSize / 3, 0, Math.PI * 2);
  ctx.stroke();
}

function getCellCoords (cell) {
  return {
     'x': (cell % 3) * cellSize,
     'y': Math.floor(cell / 3) * cellSize,
  };
}

function getCellByCoords (x, y) {
  return (Math.floor(x / cellSize) % 3) + Math.floor(y / cellSize) * 3;
}

function play (cell) {
  if (endGame) return;

  if (mapBoard[cell] != empty){
     popUpMsg();
     return;
  }

  mapBoard[cell] = humanPlayer;
  msg.textContent = "Human's player turn";//TODO: fix that

  console.log("Human Player");
  winConditions(checkWin(humanPlayer), humanPlayer);
  if (endGame == true) return;

  let currentMapBoard=[];
```

```
  for (let index = 0; index < mapBoard.length; index++) {
     if (mapBoard[index] == empty)
        currentMapBoard.push(index);
  }

  let randomO = currentMapBoard[Math.floor(Math.random() * currentMapBoard.length)];

  mapBoard[randomO]=computer;

  console.log("Computer Player");
  winConditions(checkWin(computer), computer);
  if (endGame == true) return;
}

function checkWin (player) {
  let playerMapBitMask = 0;
  console.log("playerMapBitMask: " + playerMapBitMask);
  for (let i = 0; i < mapBoard.length; i++) {
     playerMapBitMask <<= 1;
     console.log("playerMapBitMask: " + playerMapBitMask);
     if (mapBoard[i] == player){
        playerMapBitMask += 1;
        console.log("playerMapBitMask: " + playerMapBitMask);
     }
  }

  for (let i = 0; i < winPatterns.length; i++) {
     if ((playerMapBitMask & winPatterns[i]) == winPatterns[i]) {
        console.log("win Patterns: " + winPatterns[i]);
        return winPatterns[i];
     }
  }

  return 0;
}

function winConditions(winCheck,currentPlayer) {

  if (winCheck != empty) {
     endGame = true;
     msg.textContent = ((currentPlayer == X)? ' X': ' O') + '  Won';
  } else if (mapBoard.indexOf(empty) == O) {
     endGame = true;
     msg.textContent = 'Tie';
  }
```

```
  return;
}

function popUpMsg() {
  Swal.fire({
     icon: 'error',
     title: 'Oops...',
     text: 'Try another box!',
    })
}
```

5.1.1.1 Το αρχείο tictactoe.css

Στο παρακάτω μέρος του κώδικα βλέπουμε το αρχείο tictactoe.css όπου εδώ έχει γίνει το styling για το body page του tictactoe.html αλλά του κουμπιού και του popUpMsg.

```
body {
 background-color: #162243;
 font-family: "Montserrat", sans-serif;
}
button {
 background-color: brown;
 border: none;
 border-radius: 0.5rem;
 font-size: 1.5rem;
 padding: 0.7rem 0.5rem;
}
button:hover {
 cursor: pointer;
 background-color: rgb(134, 36, 36);
}
#message {
 font-size: xx-large;
 color: black;
}
```

5.1.1.2 Το αρχείο tictactoe.html

Το παρακάτω αρχείο καλεί στην αρχή του (head) την CSS / bootstrap. Στο body εμπεριέχεται το container του canvas καθώς και τις διαστάσεις που θα έχει αλλά και το restart button το οποίο το παίρνουμε από την bootstrap. Τέλος, βλέπουμε στα script tags την Javascript όπου καλείται.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Undergrade Thesis Tic Tac Toe</title>
  <!-- CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
iYQeCzEYFbKjA/T2uDLTpkwGzCiq6soy8tYaI1GyVh/UjpbCx/TYkiZhlZB6+fzT"
crossorigin="anonymous">
  <link rel="stylesheet" href="tictactoe.css">
  <!-- fonts -->
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link href="https://fonts.googleapis.com/css2?family=Montserrat&display=swap"
rel="stylesheet">

</head>
<body>
  <div class = "row">
     <div class="col-xs-12">
       <canvas id="tictactoe" width="500" height="500"></canvas>
       <p class="text-center"id="message">Welcome to Tic-Tac-Toe!</p>
       <div class="text-center">
         <button id="restart"type="button" class="btn btn-danger btn-lg">Restart</button>
     </div>
       <script type="module" src="tictactoe.js"></script>
      <script src="//cdn.jsdelivr.net/npm/sweetalert2@11"></script>
     </div>
  </div>
</body>
</html>
```

5.1.1.3 Το αρχείο index.html

Το index.html αρχείο εμπεριέχει το favicon, τον τίτλο της σελίδας, αλλά και το styling. Το body έχει μέσα το iframe όπου αυτό κεντράρετε με την βοήθεια της bootstrap στο κέντρο της σελίδας.

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
```

```
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <!-- favicon -->
  <link
    rel="apple-touch-icon"
    sizes="180x180"
    href="images/favicon_package_v0.16/apple-touch-icon.png"
  />
  <link
    rel="icon"
    type="image/png"
    sizes="32x32"
    href="images/favicon_package_v0.16/favicon-32x32.png"
  />
  <link
    rel="icon"
    type="image/png"
    sizes="16x16"
    href="images/favicon_package_v0.16/favicon-16x16.png"
  />
  <link rel="manifest" href="images/favicon_package_v0.16/site.webmanifest" />
  <link
    rel="mask-icon"
    href="images/favicon_package_v0.16/safari-pinned-tab.svg"
    color="#5bbad5"
  />
  <meta name="msapplication-TileColor" content="#da532c" />
  <meta name="theme-color" content="#ffffff" />

  <title>Tic Tac Toe Undergrade Thesis</title>
  <!-- CSS only -->
  <link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/css/bootstrap.min.css"
    rel="stylesheet"
    integrity="sha384-
iYQeCzEYFbKjA/T2uDLTpkwGzCiq6soy8tYaI1GyVh/UjpbCx/TYkiZhlZB6+fzT"
    crossorigin="anonymous"
  />
  <link rel="stylesheet" href="style.css" />
 </head>
 <body>
  <div class="row">
    <div class="container-fluid text-center mt-4">
      <iframe
        src="tictactoe.html"
```

```
        title="Tic Tac Toe"
        width="510"
        height="670"
        scrolling="no"
      ></iframe>
    </div>
  </div>
 </body>
</html>
```

5.1.1.4 Το αρχείο style.css

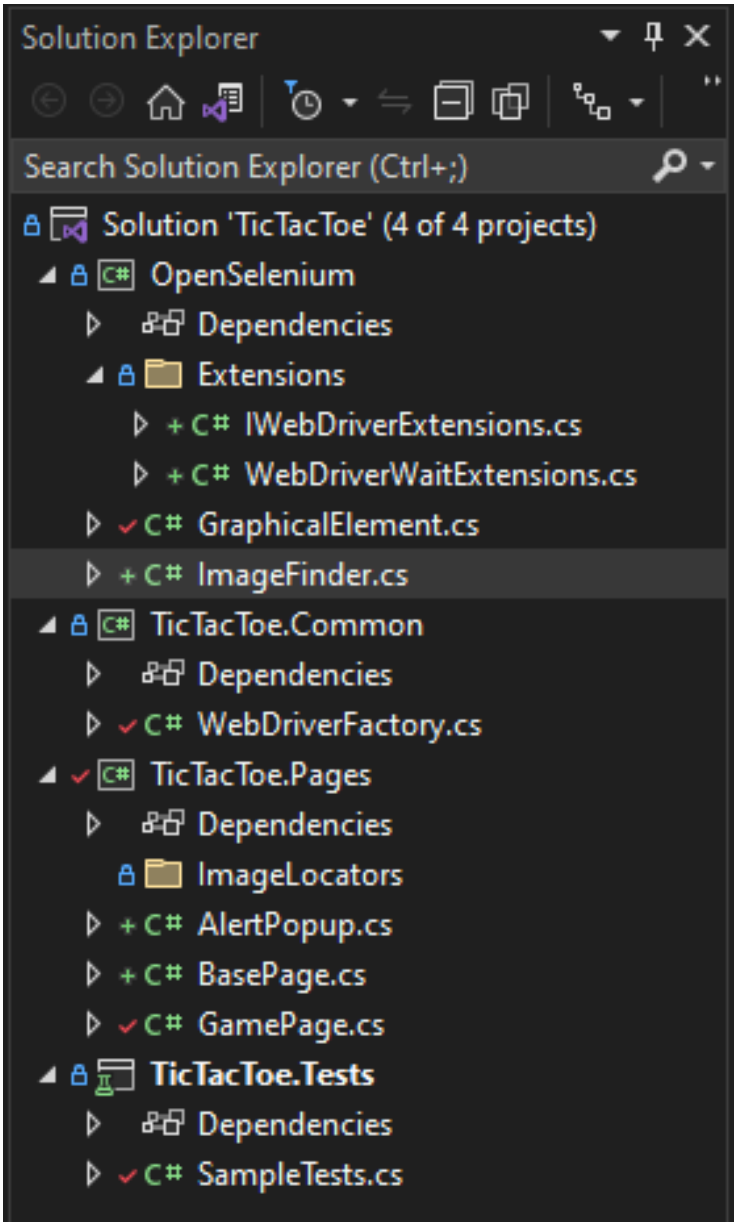
Εδώ βλέπουμε το background color που έχει το body της index.html.

```
body{
  background-color:#16213E;
}
```

5.2 Ανάπτυξη Βιβλιοθηκών και Έργου Δοκιμών

Για τη πραγμάτωση της λύσης, επιλέξαμε την δημιουργία ενός Solution με την ονομασία “TicTacToe”, το οποίο αποτελείται από τρεις βιβλιοθήκες και ένα έργο δοκιμών. Οι βιβλιοθήκες αποτελούνται από αρχεία, όπου το κάθε ένα εμπεριέχει μόνο μία κλάση. Το όνομα του κάθε αρχείου, υποδηλώνει την ονομασία της κλάσης που βρίσκεται μέσα σε αυτό.

Οι βιβλιοθήκες χωρίζονται ανάλογα με την λογική που εμπεριέχεται σε αυτές. Ξεκινώντας με μία συνοπτική εξήγηση το solution χωρίζεται στις παρακάτω βιβλιοθήκες και έργα. Στην “OpenSelenium”, όπου αναπτύξαμε τους μηχανισμούς αναγνώρισης και αλληλεπίδρασης με την εφαρμογή ιστού. Την “TicTacToe.Pages”, όπου γίνεται ένας λογικός διαχωρισμός της εφαρμογής ιστού σε κλάσεις, με την χρήση του “Page Object Model” προτύπου. Καθώς, εμπεριέχονται σε αυτή οι locators του DOM και τα πρότυπα των εικόνων. Σε αυτή την βιβλιοθήκη, γίνεται και η χρήση των μεθόδων για την εύρεση αντικειμένων του “Selenium”, όπως και οι μέθοδοι που προσφέρονται από την “ OpenSelenium” βιβλιοθήκη, για την αναγνώριση και αλληλεπίδρασή με την χρήση των προτύπων. Η “TicTacToe.Common” βιβλιοθήκη, εμπεριέχει μόνο ένα αρχείο, την “WebDriverFactory” κλάση. Σε αυτή κάνουμε χρήση του “Factory Design” σχεδιασμού. Με σκοπό την δημιουργία και την χρήση ενός μόνο αντικειμένου τύπου “IWebDriver”, από όλες τις “Page Object” κλάσεις τις βιβλιοθήκης “TicTacToe.Pages”. Εκτός από τις παραπάνω βιβλιοθήκες, μέσα στο solution υπάρχει και το έργο δοκιμών, με την ονομασία “TicTacToe.Tests”. Σε αυτό βρίσκεται η “SampleTests” κλάση. Η κλάση εμπεριέχει τις τύπου “end to end” εκτελέσιμες δοκιμές που έχουμε επιλέξει.



Εικόνα 14: Η δομή του solution.

Εν συνεχεία θα προχωρήσουμε σε μια περαιτέρω εξήγηση των μηχανισμών και της ανάπτυξης των βιβλιοθηκών, καθώς και του έργου δοκιμών.

5.2.1 Η Βιβλιοθήκη OpenSelenium

Ξεκινώντας μία πιο αναλυτική επεξήγηση για την βιβλιοθήκη “OpenSelenium”, θα αναφέρουμε τι εξαρτήσεις (dependencies) έχει, καθώς και τις κλάσεις που εμπεριέχει. Οι εξαρτήσεις της βιβλιοθήκης είναι στα εξής πακέτα: το πακέτο “OpenCvSharp.Windows”, το οποίο βρίσκεται στην έκδοση “4.5.5.20211231”, το πακέτο “Selenium.Support”, το οποίο βρίσκεται στην έκδοση “4.1.0”, και το πακέτο “Selenium.WebDriver” το οποίο βρίσκεται στην έκδοση “4.1.0”.

5.2.1.1 Εξαρτήσεις Πακέτων (Project Dependencies)

OpenSelenium

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
  </PropertyGroup>

  <ItemGroup>
      <PackageReference Include="OpenCvSharp4.Windows" Version="4.5.5.20211231" />
      <PackageReference Include="Selenium.Support" Version="4.1.0" />
      <PackageReference Include="Selenium.WebDriver" Version="4.1.0" />
  </ItemGroup>

</Project>
```

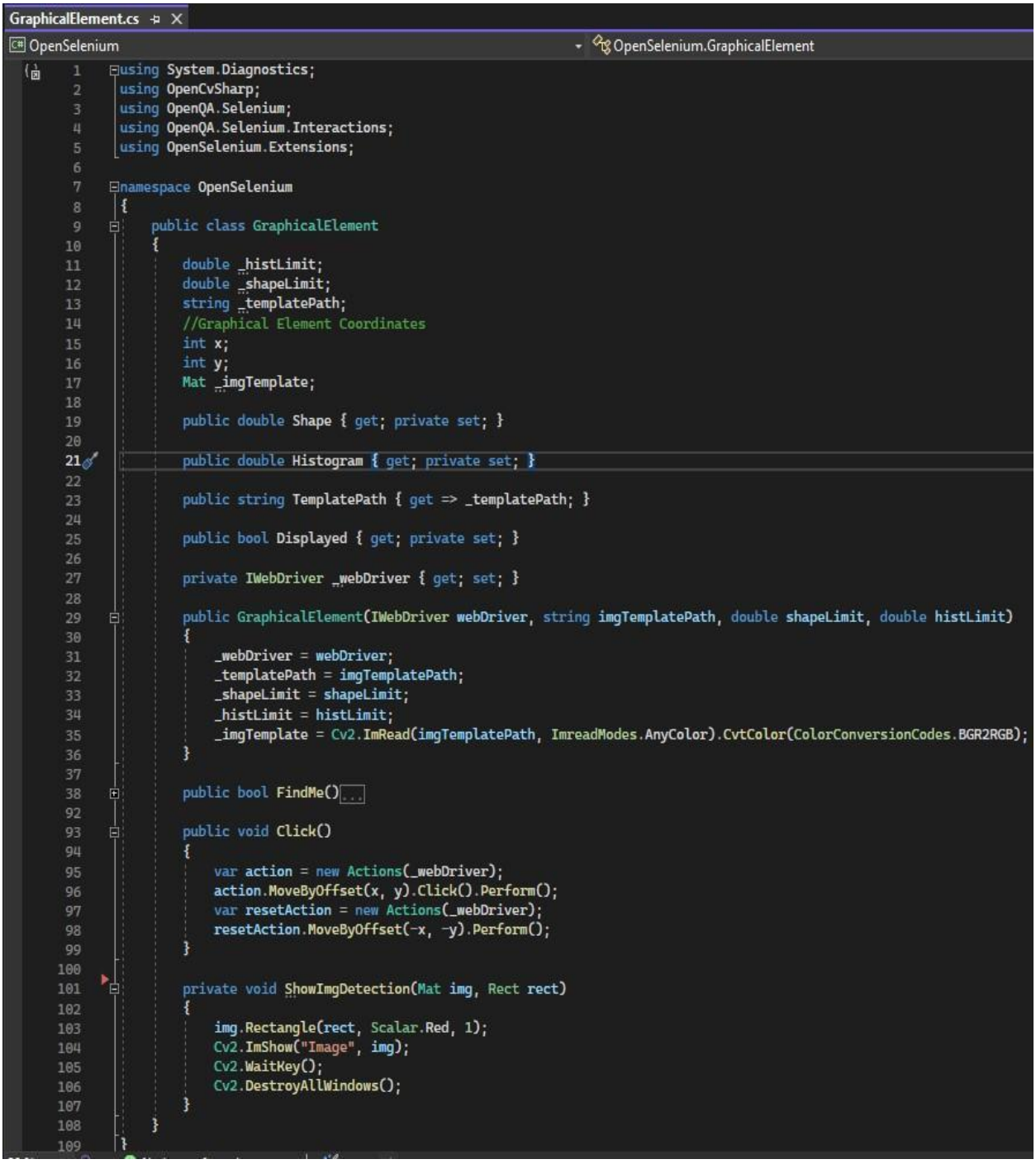
Μέσα στην βιβλιοθήκη “OpenSelenium”, βρίσκεται η το αρχείο “GraphicalElement.cs”. Κατά τη δημιουργία ενός “GraphicalElement” αντικειμένου, δίνεται σαν όρισμα μία μεταβλητή “imgTepmplatePath”, τύπου string. Πρόκειται για το path που βρίσκεται αποθηκευμένο το πρότυπο που ζητάμε από τον μηχανισμό να βρει και να αναγνωρίσει, σε μία εφαρμογή ιστού.
Επίσης για την δημιουργία ενός αντικειμένου από την εν λόγω κλάση, ζητείται να περαστεί σαν όρισμα, ένα αντικείμενο “IWebDriver”, και δύο μεταβλητές τύπου double, η “histLimit” και η “shapeLimit”. Τα τέσσερα ορίσματα θα χρησιμοποιηθούν από την μέθοδο “FindMe”.
Η μέθοδος “FindMe” θα χρησιμοποιήσει το αντικείμενο τύπου “IWebDriver”, για να τραβήξει ένα στιγμιότυπο της οθόνης, σε όλη την έκταση της σελίδας ιστού, εκείνη την χρονική στιγμή που η μέθοδος θα καλεστεί. Στην συνέχεια, με την χρήση της βιβλιοθήκης “OpenCvSharp”, θα

προσπαθήσει να εντοπίσει το πρότυπο στην οθόνη. Κατά την διαδικασία εντοπισμού του προτύπου, η μέθοδος της βιβλιοθήκης “OpenCvSharp”, πάντα εντοπίζει κάποια περιοχή πάνω στο στιγμιότυπο της οθόνης.

Ο τρόπος για να ελεγχθεί εάν η διαδικασία ήταν επιτυχής ή όχι, εξαρτάται από τις τιμές των μεταβλητών “histLimit” και η “shapeLimit”. Ορίζουμε για παράδειγμα ότι η μεταβλητή “histLimit” εμπεριέχει την τιμή 1 και η μεταβλητη “shapeLimit” την τιμή 0.92. Εάν η μέθοδος μας επιστρέψει τιμές μικρότερες των ορισμάτων που έχουμε θέσει, έστω σε μία από τις δύο μεταβλητές, τότε η αναγνώριση κρίνεται ανεπιτυχής. Εάν οι τιμές είναι ίσες ή και μεγαλύτερες των ορισμάτων, η αναγνώριση μπορεί να θεωρηθεί επιτυχείς. Με μία σύγκριση των ορισμάτων, θέτουμε και στο τύπου bool property “Displayed”, μία τιμή true ή false. Το “Displayed” property, είναι τύπου bool και είναι public, με private set, έτσι ώστε να μπορεί να οριστεί μόνο από την μέθοδο “FindMe”, αλλά να μπορεί να εκδηλώνει εάν το αντικείμενο είναι εμφανές ή και όχι.

Ένα αντίστοιχο property παρέχεται από το Selenium, στα αντικείμενα τύπου “IWebElement”. Επίσης, στην κλάση εμπεριέχονται και οι μέθοδοι “Click” και “ShowImageDetection”. Η πρώτη μας παρέχει την υλοποίηση, για να μπορεί το πρόγραμμα να αλληλοεπιδράσει με την εφαρμογή ιστού, χρησιμοποιώντας τις συντεταγμένες, στις οποίες έχει εντοπιστεί το πρότυπο. Οι συντεταγμένες έχουν βρεθεί και αποθηκευτεί σε μεταβλητές, κατά την κλήση της μεθόδου “FindMe”. Εάν το πρόγραμμα βρίσκεται σε εκτέλεση αποσφαλμάτωσης, ο μηχανικός είναι σε θέση να δει ένα παράθυρο, όπου το έχουμε ονομάσει με “Image”, να εμφανίζεται και να δείχνει τον εντοπισμό του προτύπου.

5.2.1.2 Η κλάση GraphicalElement



Το αρχείο “ ImageFinder.cs”, εμπεριέχει μόνο μία στατική μέθοδο, την “FindTemplate”. Η μέθοδος μας παρέχει τον μηχανισμό εξαγωγής ενός προτύπου, καθώς και την αποθήκευση του

στο Temp φάκελο του λειτουργικού συστήματος Windows. Κατά την χρήση του ο μηχανικός χρειάζεται να περιηγηθεί στην εφαρμογή ιστού μέχρι να φτάσει στην κατάσταση όπου θέλει να εξαγάγει το πρότυπο. Στην συνέχεια, κάνει κλήση της μεθόδου, και ένα παράθυρο με το στιγμιότυπο της οθόνης εμφανίζεται μπροστά του. Κάνοντας “crop” στην εικόνα και πατώντας το πλήκτρο “Enter”, το πρότυπο αποθηκεύεται.

5.2.1.3 Η κλάση ImageFinder

ImageFinder.cs

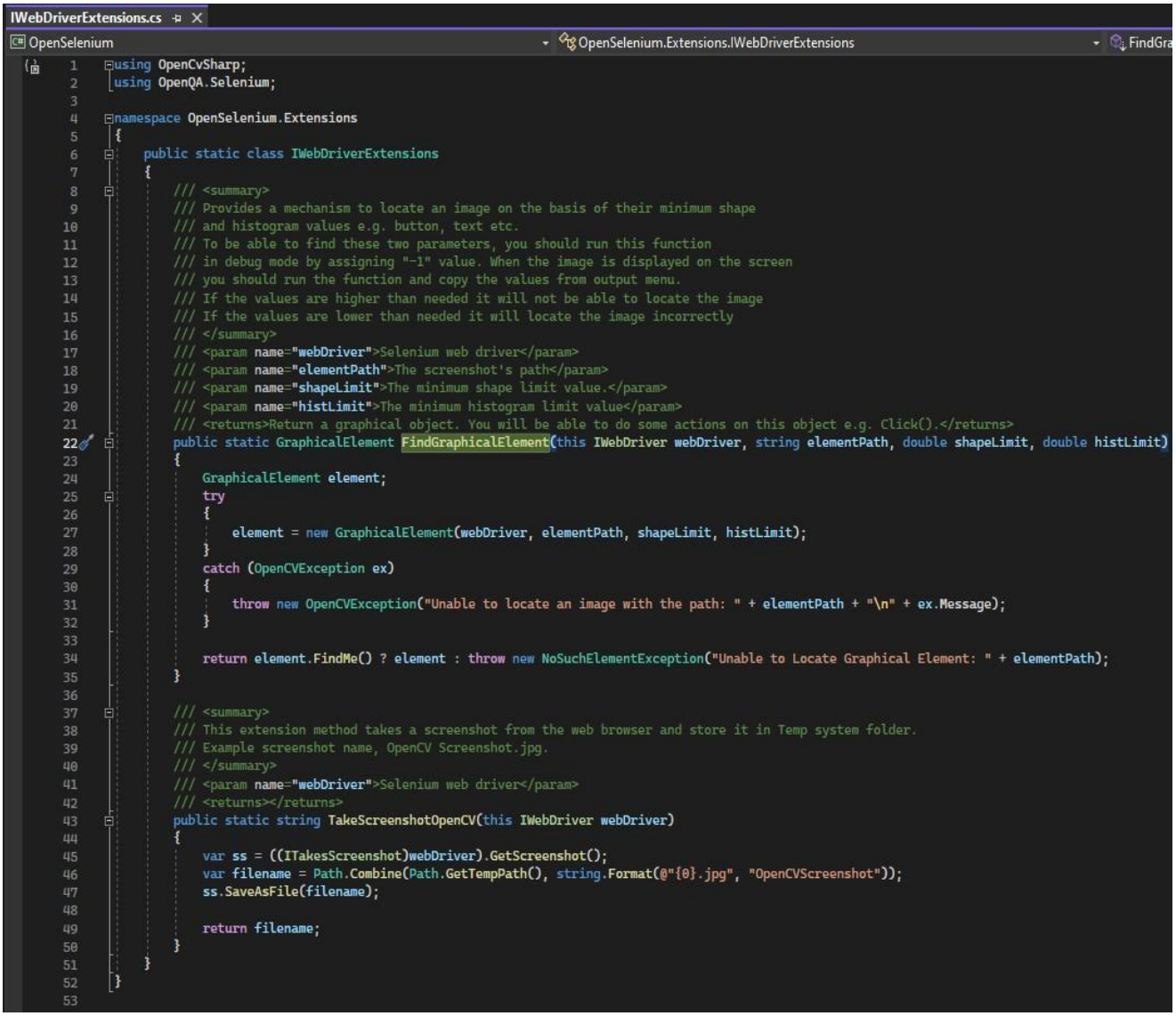
OpenSelenium — OpenSelenium.ImageFinder

```
using System.Diagnostics;
using OpenCvSharp;
using OpenQA.Selenium;
using OpenSelenium.Extensions;

namespace OpenSelenium
{
    public static class ImageFinder
    {
        static int templateNum = 17;

        /// <summary>
        /// Detect and save an image template.
        /// This method can be used only in Debug mode.
        /// </summary>
        /// <param name="webDriver">Selenium web driver.</param>
        /// <param name="elementPath">The new image path.</param>
        /// <exception cref="InvalidOperationException"></exception>
        /// <exception cref="OpenCVException"></exception>
        public static string FindTemplate(IWebDriver webDriver)
        {
            if (!Debugger.IsAttached)
                throw new InvalidOperationException("This method can be used only when the Debugger is attached.");

            var screenshotPath = webDriver.TakeScreenshotOpenCV();
            Mat screenShotImg = Cv2.ImRead(screenshotPath);
            var templateRect = Cv2.SelectROI("Screenshot", screenShotImg);
            var templateImg = new Mat(screenShotImg, templateRect);

            Cv2.ImShow("Screenshot", screenShotImg);
            Cv2.ImShow("Template", templateImg);
            Cv2.WaitKey();
            Cv2.DestroyAllWindows();

            templateNum++;
            var fileName = $"template{ templateNum }";
            var filePath = Path.Combine(Path.GetTempPath(), string.Format("{0}.jpg", fileName));

            bool isSaved = Cv2.ImWrite(filePath, templateImg);

            if (!isSaved)
                throw new Exception($"Path { filePath } is not reachable.");

            Debug.WriteLine($"Folder Path: { Path.Combine(Path.GetTempPath()) }\n" +
                            $"File Name: { fileName }");

            return filePath;
        }
    }
}
```

Στον φάκελο “Extensions”, βρίσκονται τα αρχεία “IWebDriverExtensions.cs” και “WebDriverWaitExtensions.cs”. Στο πρώτο εντοπίζουμε extensions του “IWebDriver”. Πρόκειται για την μέθοδο “TakeScreenshot”, που χρησιμοποιείται για να τραβήξουμε ένα στιγμιότυπο της εφαρμογής ιστού, με το όνομα “OpenCVScreeshot”. Η χρήση αυτής της μεθόδου

γίνεται κατά την εξαγωγή ενός προτύπου, όπως και κατά την διαδικασία αναγνώρισης. Η δεύτερη extension μέθοδος είναι η “FindGraphicalElement”. Πρόκειται για την μέθοδο που χρησιμοποιεί την “FindMe”, και αν το πρότυπο έχει εντοπιστεί, η μέθοδος επιστρέφει ένα αντικείμενο τύπου “GraphicalElement”. Στην περίπτωση που δεν καταφέρει να εντοπίσει το μονοπάτι, που βρίσκεται το πρότυπο, εμφανίζεται ένα exception τύπου “OpenCVException”. Εάν δεν εντοπιστεί το πρότυπο, την συγκεκριμένη χρονική στιγμή, στο στιγμιότυπο της οθόνης. Η μέθοδος εμφανίζεται ένα exception τύπου “NoSuchElementException”. Επίσης, τα exception εμφανίζουν τα κατάλληλα μηνύματα, ώστε να είναι πιο εύχρηστα σε μία διαδικασία αποσφαλμάτωσης, από τον μηχανικό.

5.2.1.4 Η κλάση IWebDriverExtensions

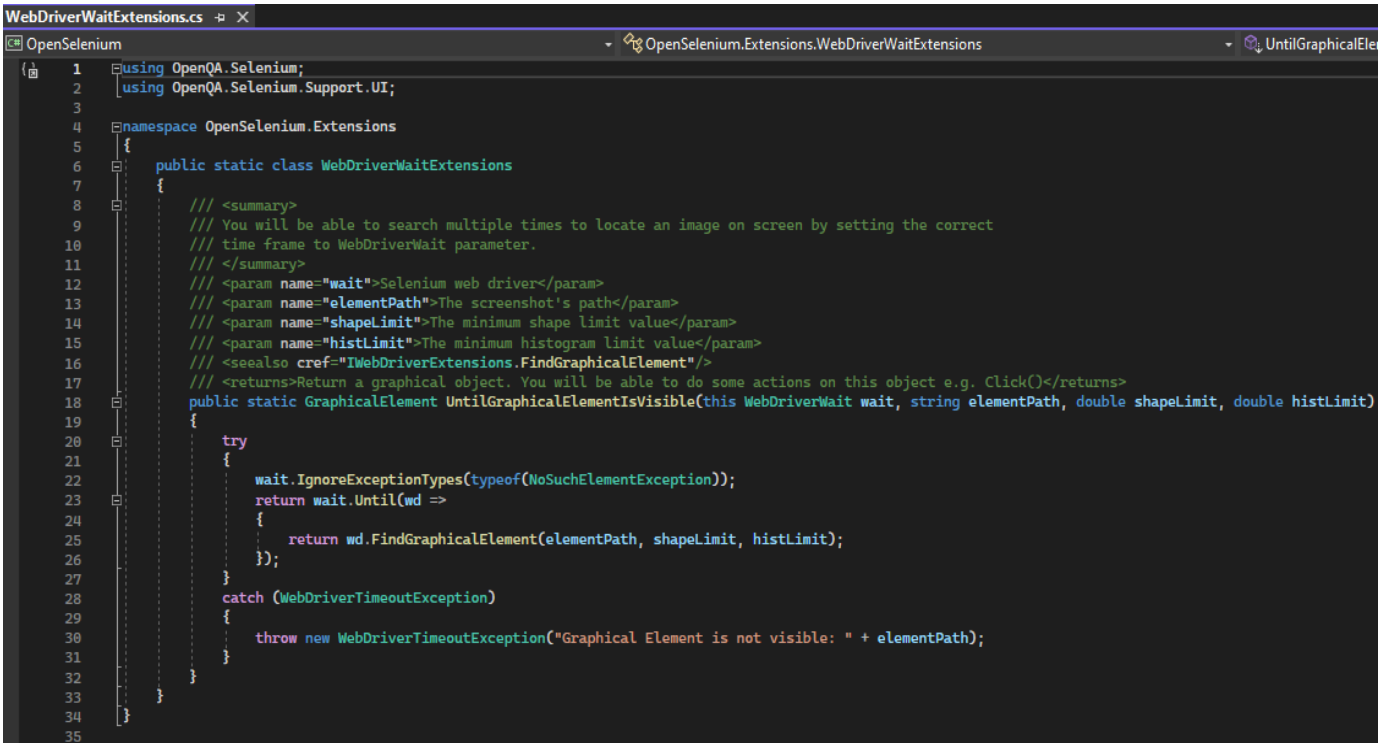


```
using OpenCvSharp;
using OpenQA.Selenium;

namespace OpenSelenium.Extensions
{
    public static class IWebDriverExtensions
    {
        /// <summary>
        /// Provides a mechanism to locate an image on the basis of their minimum shape
        /// and histogram values e.g. button, text etc.
        /// To be able to find these two parameters, you should run this function
        /// in debug mode by assigning "-1" value. When the image is displayed on the screen
        /// you should run the function and copy the values from output menu.
        /// If the values are higher than needed it will not be able to locate the image
        /// If the values are lower than needed it will locate the image incorrectly
        /// </summary>
        /// <param name="webDriver">Selenium web driver</param>
        /// <param name="elementPath">The screenshot's path</param>
        /// <param name="shapeLimit">The minimum shape limit value.</param>
        /// <param name="histLimit">The minimum histogram limit value</param>
        /// <returns>Return a graphical object. You will be able to do some actions on this object e.g. Click().</returns>
        public static GraphicalElement FindGraphicalElement(this IWebDriver webDriver, string elementPath, double shapeLimit, double histLimit)
        {
            GraphicalElement element;
            try
            {
                element = new GraphicalElement(webDriver, elementPath, shapeLimit, histLimit);
            }
            catch (OpenCVException ex)
            {
                throw new OpenCVException("Unable to locate an image with the path: " + elementPath + "\n" + ex.Message);
            }

            return element.FindMe() ? element : throw new NoSuchElementException("Unable to Locate Graphical Element: " + elementPath);
        }

        /// <summary>
        /// This extension method takes a screenshot from the web browser and store it in Temp system folder.
        /// Example screenshot name, OpenCV Screenshot.jpg.
        /// </summary>
        /// <param name="webDriver">Selenium web driver</param>
        /// <returns></returns>
        public static string TakeScreenshotOpenCV(this IWebDriver webDriver)
        {
            var ss = ((ITakesScreenshot)webDriver).GetScreenshot();
            var filename = Path.Combine(Path.GetTempPath(), string.Format(@"{0}.jpg", "OpenCVScreenshot"));
            ss.SaveAsFile(filename);

            return filename;
        }
    }
}
```

Στο δεύτερο αρχείο “WebDriverWaitExtensions.cs”, βρίσκεται η αντίστοιχη στατική κλάση. Σε αυτή, εντοπίζουμε την extension μέθοδο “UntilGraphicalElementIsVisible”. Είναι μία extension μέθοδος που καλείται από ένα αντικείμενο τύπου “WebDriverWait”. Χρησιμοποιείται η μέθοδος “Until”, όπου χρησιμεύει για να γίνεται μία διαρκής χρήση της μεθόδου “FindGraphicalElement”,για την αναζήτηση του προτύπου. Το αντικείμενο “WebDriverWait”, κρατάει σε ένα property, το ορισμένο κατά την δημιουργία του χρονικό διάστημα, κατά το οποίο πραγματοποιείται η αναζήτηση. Σε περίπτωση που δεν επιτευχθεί η αναγνώριση του προτύπου,

πιάνουμε το exception τύπου “WebDriverTimeoutException”, και το εμφανίζουμε ξανά με ένα πιο επεξηγηματικό μήνυμα.

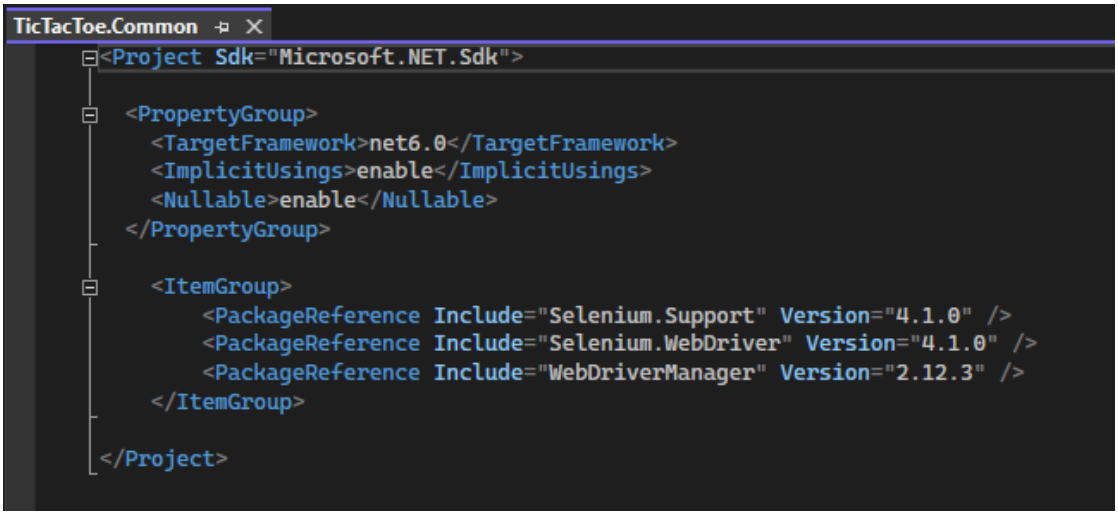
5.2.1.5 Η κλάση WebDriverWaitExtensions



5.2.1.6 Η βιβλιοθήκη TicTacToe.Common

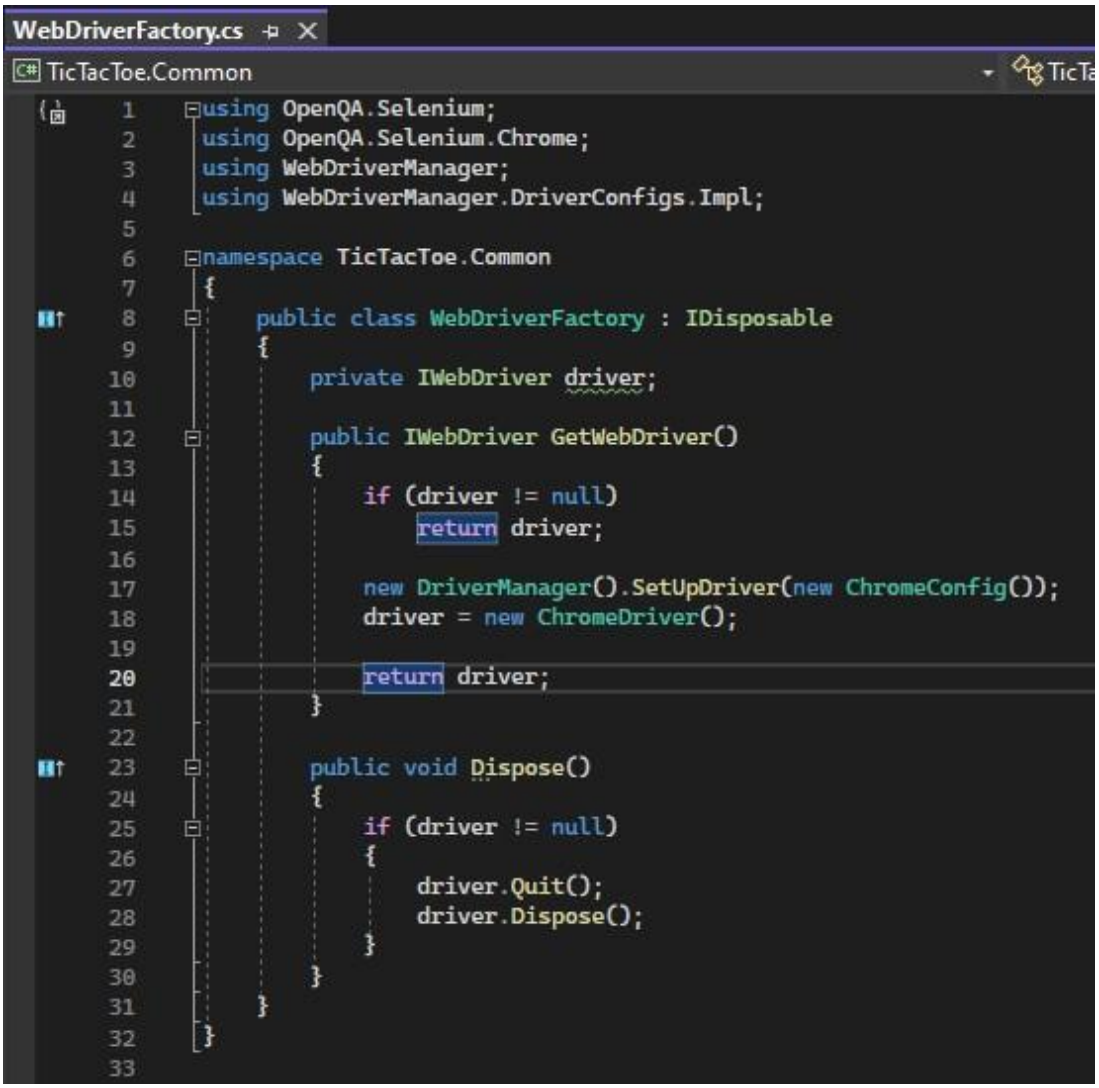
Στο έργο βρίσκουμε την βιβλιοθήκη με την ονομασία “TicTacToe.Common”. Οι εξαρτήσεις τις βιβλιοθήκης είναι στα εξής πακέτα: το πακέτο “WebDriverManager”, το οποίο βρίσκεται στην έκδοση “2.12.3”, το πακέτο “Selenium.Support”, το οποίο βρίσκεται στην έκδοση “4.1.0”, και το πακέτο “Selenium.WebDriver” το οποίο βρίσκεται στην έκδοση “4.1.0”.

5.2.1.7 Εξαρτήσεις Πακέτων (Project Dependencies)



Είναι μία ξεχωριστή βιβλιοθήκη που εμπεριέχει μόνο το αρχείο “WebDriverFactory.cs”, με την αντίστοιχη κλάση. Είναι ένα ξεχωριστό έργο μέσα στο solution, όπου υλοποιούμε το “Factory Design Pattern”. Καθώς και όποια αντικείμενα “WebDriverFactory” δημιουργηθούν, θα είναι Disposable, καθώς η κλάση υλοποιεί το interface “IDisposable”.

5.2.1.8 Η κλάση WebDriverFactory



5.2.1.9 Η βιβλιοθήκη TicTacToe.Pages

Το έργο “TicTacToe.Pages” εμπεριέχει αμιγώς τα κομμάτια κώδικα, και τις υλοποιήσεις ώστε να παρέχονται σε κάθε έργο δοκιμών, που ενδέχεται να προστεθεί στο solution, ξεχωριστά αντικείμενα με βάση τον διαχωρισμό που έχει γίνει για την εκάστοτε σελίδα ιστού. Σκοπός είναι να γνωρίζει μόνο η κάθε κλάση πως θα εντοπίσει και θα αλληλοεπιδράσει με κάθε αντικείμενο του DOM, μέσω δημοσίων μεθόδων του Selenium ή του έργου “OpenSelenium”, με την χρήση

αναγνώρισης εικόνας. Επίσης αξίζει να αναφερθεί ότι για την επίτευξη του επιτυχούς διαχωρισμού του παιχνιδιού σε κλάσεις όπως η “GamePage”, “AlertPopup”, έγινε χρήση του “Page Object Pattern”. Επιπροσθέτως, η χρήση του συγκεκριμένου προτύπου υλοποίησης, μας επιτρέπει την ύπαρξη μίας μόνο, κοινής βάσης κώδικα του Selenium.

Οι εξαρτήσεις τις βιβλιοθήκης είναι στα εξής πακέτα: το πακέτο “DotNetSeleniumExtras.WaitHelpers”, το οποίο βρίσκεται στην έκδοση “3.11.0”, το πακέτο “Selenium.Support”, το οποίο βρίσκεται στην έκδοση “4.1.0”, και το πακέτο “Selenium.WebDriver” το οποίο βρίσκεται στην έκδοση “4.1.0”.

5.2.1.10 Εξαρτήσεις Πακέτων (Project Dependencies)

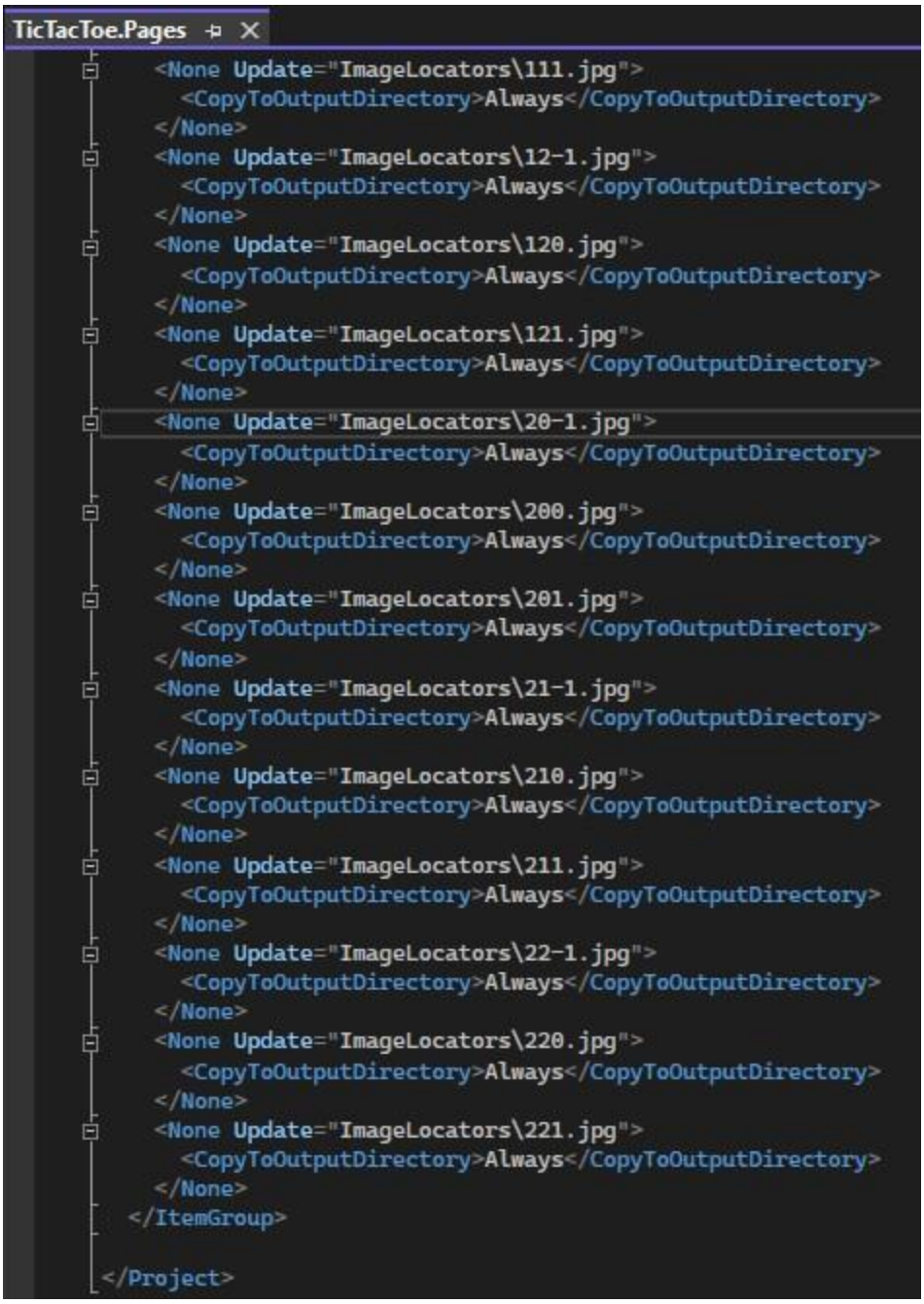
TicTacToe.Pages

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Selenium.Support" Version="4.1.0" />
    <PackageReference Include="Selenium.WebDriver" Version="4.1.0" />
    <PackageReference Include="DotNetSeleniumExtras.WaitHelpers" Version="3.11.0" />
  </ItemGroup>

  <ItemGroup>
    <ProjectReference Include="..\OpenSelenium\OpenSelenium.csproj" />
  </ItemGroup>
```

5.2.1.11 Image Locators

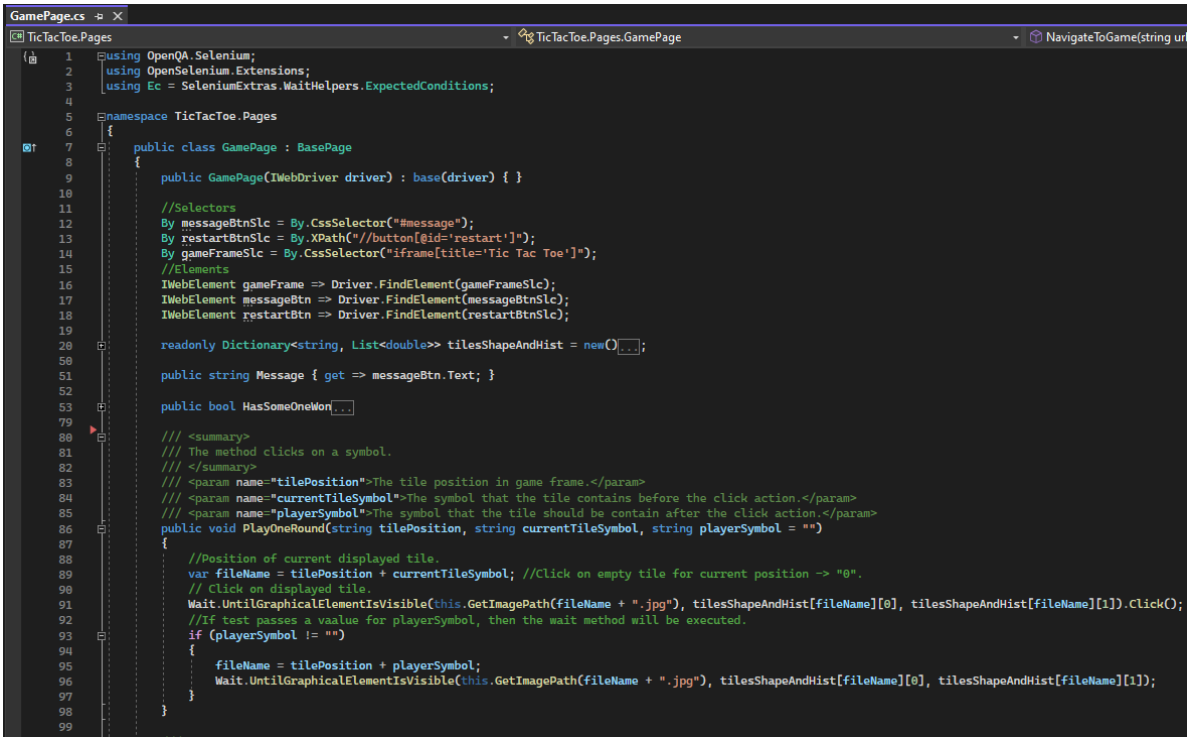
TicTacToe.Pages

```
<ItemGroup>
  <None Update="ImageLocators\00-1.jpg">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </None>
  <None Update="ImageLocators\000.jpg">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </None>
  <None Update="ImageLocators\001.jpg">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </None>
  <None Update="ImageLocators\01-1.jpg">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </None>
  <None Update="ImageLocators\010.jpg">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </None>
  <None Update="ImageLocators\011.jpg">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </None>
  <None Update="ImageLocators\02-1.jpg">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </None>
  <None Update="ImageLocators\020.jpg">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </None>
  <None Update="ImageLocators\021.jpg">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </None>
  <None Update="ImageLocators\10-1.jpg">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </None>
  <None Update="ImageLocators\100.jpg">
    <CopyToOutputDirectory>Always</CopyToOutputDirectory>
  </None>
```

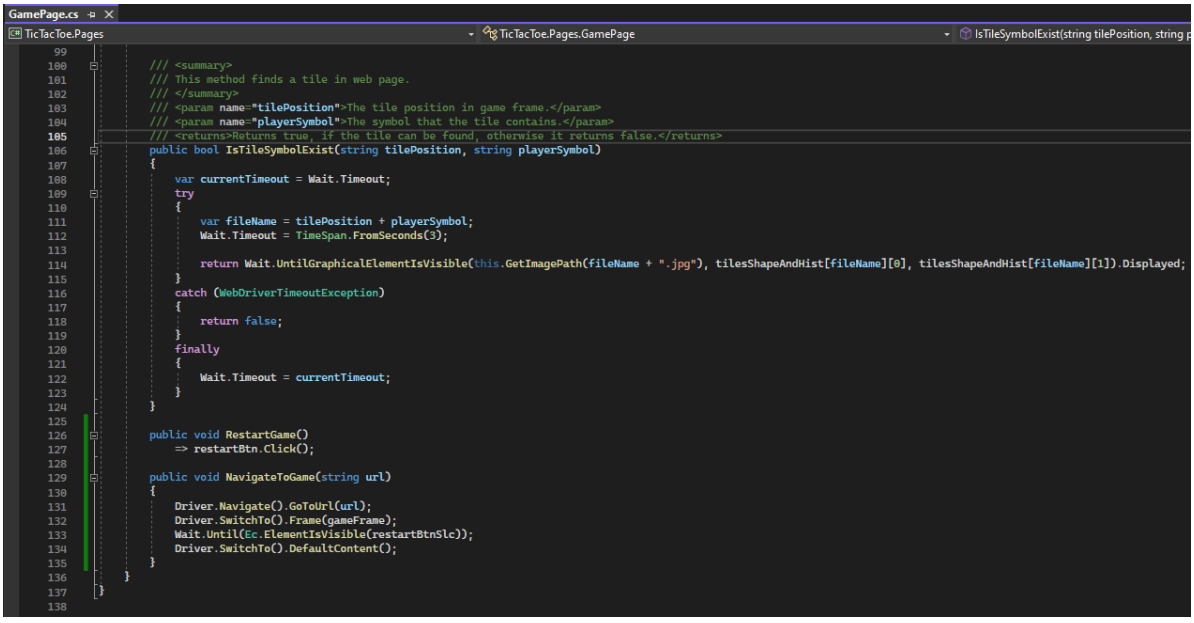


Στο φάκελο με την ονομασία “ImageLocators” βρίσκονται τα πρότυπα των εικόνων της εφαρμογής ιστού, όπου χρησιμοποιούνται στην κλάση “GamePage”, με παρόμοιο τρόπο όπως οι locators που βρίσκουμε στο DOM, από τις μεθόδους “FindGraphicalElement” και “UntilGraphicalElementIsVisible ”.

5.2.1.12 Η κλάση GamePage



```
using OpenQA.Selenium;
using OpenSelenium.Extensions;
using Ec = SeleniumExtras.WaitHelpers.ExpectedConditions;

namespace TicTacToe.Pages
{
    public class GamePage : BasePage
    {
        public GamePage(IWebDriver driver) : base(driver) { }

        //Selectors
        By messageBtnSlc = By.CssSelector("#message");
        By restartBtnSlc = By.XPath("//button[@id='restart']");
        By gameFrameSlc = By.CssSelector("iframe[title='Tic Tac Toe']");
        //Elements
        IWebElement gameFrame => Driver.FindElement(gameFrameSlc);
        IWebElement messageBtn => Driver.FindElement(messageBtnSlc);
        IWebElement restartBtn => Driver.FindElement(restartBtnSlc);

        readonly Dictionary<string, List<double>> tilesShapeAndHist = new()...;

        public string Message { get => messageBtn.Text; }

        public bool HasSomeOneWon...

        /// <summary>
        /// The method clicks on a symbol.
        /// </summary>
        /// <param name="tilePosition">The tile position in game frame.</param>
        /// <param name="currentTileSymbol">The symbol that the tile contains before the click action.</param>
        /// <param name="playerSymbol">The symbol that the tile should be contain after the click action.</param>
        public void PlayOneRound(string tilePosition, string currentTileSymbol, string playerSymbol = "")
        {
            //Position of current displayed tile.
            var fileName = tilePosition + currentTileSymbol; //Click on empty tile for current position -> "0".
            // Click on displayed tile.
            Wait.UntilGraphicalElementIsVisible(this.GetImagePath(fileName + ".jpg"), tilesShapeAndHist[fileName][0], tilesShapeAndHist[fileName][1]).Click();
            //If test passes a vaalue for playerSymbol, then the wait method will be executed.
            if (playerSymbol != "")
            {
                fileName = tilePosition + playerSymbol;
                Wait.UntilGraphicalElementIsVisible(this.GetImagePath(fileName + ".jpg"), tilesShapeAndHist[fileName][0], tilesShapeAndHist[fileName][1]);
            }
        }
```



```
        /// <summary>
        /// This method finds a tile in web page.
        /// </summary>
        /// <param name="tilePosition">The tile position in game frame.</param>
        /// <param name="playerSymbol">The symbol that the tile contains.</param>
        /// <returns>Returns true, if the tile can be found, otherwise it returns false.</returns>
        public bool IsTileSymbolExist(string tilePosition, string playerSymbol)
        {
            var currentTimeout = Wait.Timeout;
            try
            {
                var fileName = tilePosition + playerSymbol;
                Wait.Timeout = TimeSpan.FromSeconds(3);

                return Wait.UntilGraphicalElementIsVisible(this.GetImagePath(fileName + ".jpg"), tilesShapeAndHist[fileName][0], tilesShapeAndHist[fileName][1]).Displayed;
            }
            catch (WebDriverTimeoutException)
            {
                return false;
            }
            finally
            {
                Wait.Timeout = currentTimeout;
            }
        }

        public void RestartGame()
            => restartBtn.Click();

        public void NavigateToGame(string url)
        {
            Driver.Navigate().GoToUrl(url);
            Driver.SwitchTo().Frame(gameFrame);
            Wait.Until(Ec.ElementIsVisible(restartBtnSlc));
            Driver.SwitchTo().DefaultContent();
        }
    }
}
```

5.2.1.13 Η κλάση AlertPopup

AlertPopup.cs

TicTacToe.Pages — TicTacToe.Pages.AlertPopup

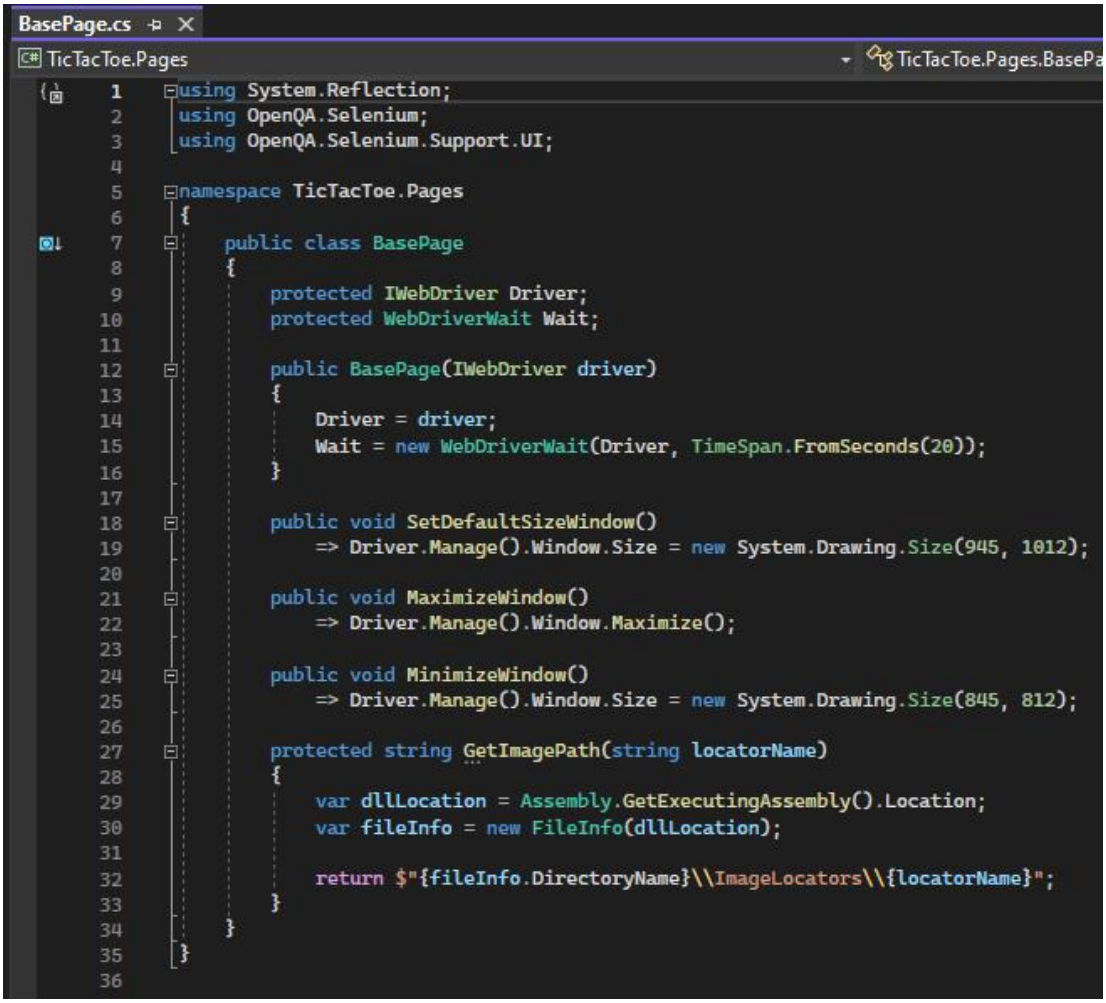
```
using OpenQA.Selenium;
using Ec = SeleniumExtras.WaitHelpers.ExpectedConditions;

namespace TicTacToe.Pages
{
    public class AlertPopup : BasePage
    {
        public AlertPopup(IWebDriver driver) : base(driver) { }

        //Selectors
        By messageSlc => By.CssSelector("#swal2-html-container");
        By gameFrameSlc => By.CssSelector("iframe[title='Tic Tac Toe']");
        By acceptBntSlc => By.CssSelector("button.swal2-confirm.swal2-styled");
        //Elements
        IWebElement gameFrame => Driver.FindElement(gameFrameSlc);
        IWebElement message => Wait.Until(Ec.ElementIsVisible(messageSlc));
        IWebElement acceptBnt => Wait.Until(Ec.ElementToBeClickable(acceptBntSlc));

        public string Message
        {
            get
            {
                Driver.SwitchTo().Frame(gameFrame);
                var messageTxt = message.Text;
                Driver.SwitchTo().ParentFrame();

                return messageTxt;
            }
        }

        public bool IsVisible
        {
            get
            {
                try
                {
                    Driver.SwitchTo().Frame(gameFrame);

                    return acceptBnt.Displayed;
                }
                catch (WebDriverTimeoutException)
                {
                    return false;
                }
                finally
                {
                    Driver.SwitchTo().ParentFrame();
                }
            }
        }

        public void Accept()
            => acceptBnt.Click();
    }
}
```

Η κλάση “AlertPopup”, εμπεριέχει αμιγώς μεθόδους του Selenium και δεν γίνεται χρήση των μεθόδων αναγνώρισης, μέσα σε αυτή.

Θα αναφέρουμε επίσης, την κλάση “BasePage”. Σε αυτή, χρησιμοποιούμε έναν κοινό “constructor” για όλα τα αντικείμενα. Επίσης βρίσκουμε και κοινές δημόσιες μεθόδους, για να αλλάζουμε μέσα από το έργο δοκιμών, το μέγεθος του παραθύρου του περιηγητή, καθώς και την

protected μέθοδο “GetImagePath”, η οποία βρίσκει και επιστρέφει την τοποθεσία του προτύπου μέσα στη βιβλιοθήκη, σε μορφή string.

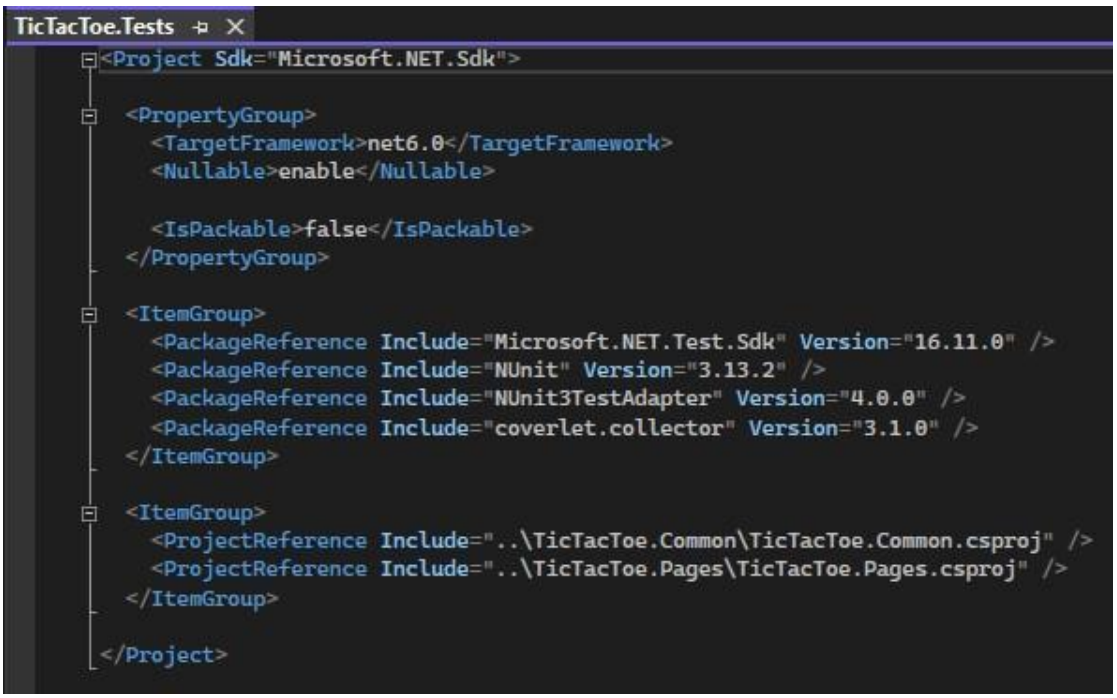
5.2.1.14 Η κλάση BasePage



5.2.1.15 Το έργο δοκιμών TicTacToe.Tests

Στο έργο δοκιμών “TicTacToe.Tests”, φιλοξενούνται οι δοκιμές για το παιχνίδι ιστού που έχουμε αναπτύξει. Οι εξαρτήσεις του έργου δοκιμών είναι στα εξής πακέτα: το πακέτο “Microsoft.NET.test.Sdk”, το οποίο βρίσκεται στην έκδοση “16.11.0”, το πακέτο “NUnit”, το οποίο βρίσκεται στην έκδοση “3.13.2”, το πακέτο “NUnit3TestAdapter”, το οποίο βρίσκεται στην έκδοση “4.0.0”, και το πακέτο “coverlet.collector” το οποίο βρίσκεται στην έκδοση “3.1.0”.

5.2.1.16 Εξαρτήσεις Πακέτων (Project Dependencies)



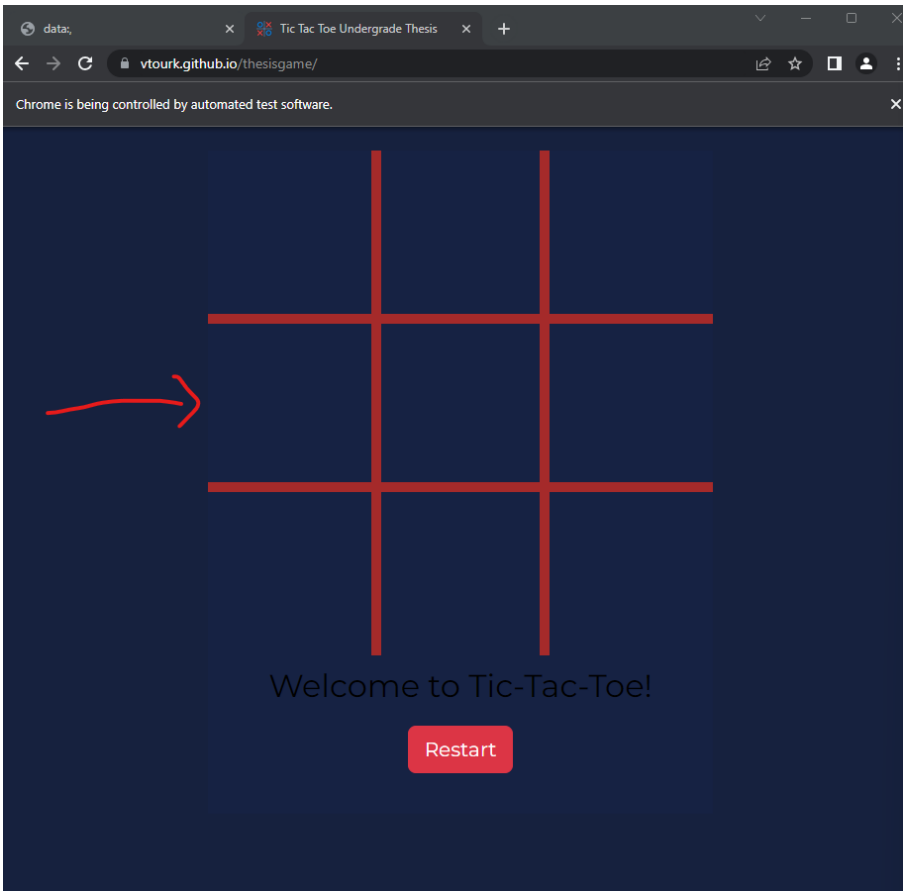
Μέσα σε αυτό βρίσκουμε το αρχείο “SampleTests.cs”, με την κλάση “SampleTests”, όπου φιλοξενούνται οι δοκιμές. Μέσα σε αυτή έχουμε τοποθετήσει τις μεθόδους “BeforeTestRun”, “BeforeTestExecution”, “AfterTestRun” και “ AfterTestExecution”. Πρόκειται για τις μεθόδους, που όπως δηλώνει η ονομασία του, τρέχουν πριν από την εκτέλεση όλων των δοκιμών, ανοίγοντας ένα πρόγραμμα περιήγησης, έχουμε επιλέξει τον περιηγητή Google Chrome, αλλά οι δοκιμές θα μπορούσαν να εκτελεστούν και σε άλλα προγράμματα περιήγησης όπως το Mozilla Firefox.

Πριν από κάθε δοκιμή, ανοίγουμε μία καινούργια σελίδα περιήγησης και μεταφερόμαστε μέσω της διεύθυνσης που φιλοξενούμε την εφαρμογή ιστού, στο παιχνίδι της τρίλιζας. Επίσης επιλέγουμε να προσθέσουμε στην ήδη αρχικοποιημένη λίστα “gameTiles”, τις εννέα θέσεις με την τιμή μηδέν. Το μηδέν αναφέρεται στα κενά κουτιά, η τιμή ένα στα κουτιά που έχει επιλέξει η δοκιμή να τοποθετήσει το σύμβολο “X” και η τιμή πλην ένα στα κουτιά που έχει επιλέξει το παιχνίδι να τοποθετήσει το σύμβολο “O”. Εάν η λίστα εμπεριέχει ήδη θέσεις από εκτελέσεις προηγούμενων δοκιμών, τότε αφαιρούμαι ότι εμπεριέχει και την ετοιμάζουμε για την επόμενη εκτέλεση της δοκιμής. Αξίζει να αναφερθεί στο σημείο αυτό, ότι σε περίπτωση παράλληλης

εκτέλεσης των δοκιμών, θα πρέπει να γίνουν αλλαγές, και μία από αυτές θα έπρεπε να είναι η αρχικοποίηση και διαχείριση της λίστα, μέσα στην εκάστοτε δοκιμή.

Μετά από την εκτέλεση κάθε δοκιμής, η σελίδα του παιχνιδιού κλείνει. Ενώ μετά την εκτέλεση όλων των δοκιμών, καλείται η μέθοδος “Dispose” του αντικειμένου τύπου “WebDriverFactory”, η οποία κλείνει το πρόγραμμα περιήγησης. Μέσα στην κλάση δοκιμών, υπάρχουν τρείς ιδιωτικές μέθοδοι. Η πρώτη είναι η “FindAllAvailableGameTiles”, όπου χρησιμοποιείται για να βρεθούν όλες οι κενές θέσεις της λίστα “gameTiles”, δηλαδή όλα τα κενά κουτιά του παιχνιδιού. Η μέθοδος “FindRendomPlayableTile”, όπου βρίσκει όλες τις κενές θέσεις τις λίστας “gameTiles”, και μας επιλέγει και μας επιστρέφει μέσα στην δοκιμή, μία τυχαία κενή θέση, για να τοποθετηθεί το σύμβολο “X”. Η τελευταία από τις ιδιωτικές μεθόδους, είναι η μέθοδος “FindTilePosition”. Η οποία βάση τις θέσεις τύπου int της λίστας “gameTiles”,που περνάμε παραμετρικά, μας επιστρέφει μία μεταβλητή τύπου string. Η μεταβλητή αποτελείται από δύο char χαρακτήρες. Η κάθε θέση της λίστας, από τις εννέα θέσεις, μεταφράζεται σε ένα από τα κουτιά της τρίλιζας.

Το πρώτο και το δεύτερο char της μεταβλητής string, μπορεί να είναι από μηδέν έως δύο. Ο πρώτος χαρακτήρας υποδηλώνει την σειρά, και ο δεύτερος την στήλη. Για παράδειγμα, το μηδέν είναι η πρώτη σειρά της τρίλιζας, το ένα η δεύτερη και το δύο η τρίτη. Αντίστοιχα, για τον δεύτερο χαρακτήρα char, το μηδέν είναι η πρώτη στήλη της τρίλιζας, το ένα η δεύτερη και το δύο η τρίτη. Η μεταβλητή string που μας επιστέφει αυτή η μέθοδος, είναι που χρησιμοποιούμε για να περάσουμε στην μέθοδο “PlayOneRound” του αντικειμένου τύπου “GamePage”, μαζί με την μεταβλητή “emptyTile”. Δίνοντας μία μεταβλητή με την σειρά, την στήλη και το εάν είναι κενό ή πιο σύμβολο εμπεριέχεται στο κουτί, φτιάχνουμε για παράδειγμα μία μεταβλητή “100”. Δηλαδή, το πρώτο κουτί της δεύτερης σειράς, όπως φαίνεται στην εικόνα παρακάτω.



Με αυτόν τον τρόπο, ορίζουμε πιο πρότυπο, θα ψάξει να βρει η μέθοδος “PlayOneRound”, και να αλληλοεπιδράσει μαζί του. Γνωρίζοντας, την θέση και εάν είναι κενό ή πιο σύμβολο βρίσκεται εκείνη την στιγμή στο κουτί, μπορούμε η μέθοδος μπορεί να αλληλοεπιδράσει με κουτιά που δεν έχουν κάποιο σύμβολο, αλλά και με κουτιά που εμπεριέχουν.

5.2.1.17 Η κλάση SampleTests (Μέθοδοι που εκτελούνται πριν και μετά, από κάθε δοκιμή)

SampleTests.cs

TicTacToe.Tests

TicTacToe.Tests.Sa

```
1   using System;
2   using System.Collections.Generic;
3   using OpenQA.Selenium;
4   using NUnit.Framework;
5   using TicTacToe.Pages;
6   using TicTacToe.Common;
7   using System.Diagnostics;
8
9   namespace TicTacToe.Tests
10  {
11      public class SampleTests
12      {
13          IWebDriver _driver;
14          GamePage _gamePage;
15          WebDriverFactory webDriverFactory;
16          //Game logic
17          readonly int xTile = 1;
18          readonly int oTile = -1;
19          readonly int emptyTile = 0;
20          private List<int> gameTiles = new();
21
22          [OneTimeSetUp]
23          public void BeforeTestRun()
24          {
25              webDriverFactory = new WebDriverFactory();
26              _driver = webDriverFactory.GetWebDriver();
27          }
28
29          [SetUp]
30          public void BeforeTestExecution()
31          {
32              _driver.SwitchTo().NewWindow(WindowType.Tab);
33              //Initialize page objects
34              _gamePage = new GamePage(_driver);
35              _gamePage.NavigateToGame("https://vtourk.github.io/thesisgame");
36
37              if (gameTiles.Count != 0)
38                  gameTiles.Clear();
39
40              for (int i = 1; i < 10; i++)
41              {
42                  gameTiles.Add(0);
43              }
44          }
45
46          [TearDown]
47          public void AfterTestExecution()
48          {
49              _driver.Close();
50              _driver.SwitchTo().Window(_driver.WindowHandles[0]);
51          }
52
53          [OneTimeTearDown]
54          public void AfterTestRun()
55              => webDriverFactory.Dispose();
```

5.2.1.18 Η κλάση SampleTests (Ιδιωτικές μέθοδοι)

```
163         private List<int> FindAllAvailableGameTiles(List<int> currentTiles)
164         {
165             List<int> availablePositions = new();
166 
167             for (int i = 0; i < currentTiles.Count; i++)
168             {
169                 if (currentTiles[i] == 0)
170                     availablePositions.Add(i);
171             }
172 
173             return availablePositions;
174         }
175 
176         private int FindRandomPlayableTile(List<int> currentTiles)
177         {
178             List<int> availablePositions = new();
179 
180             for (int i = 0; i < currentTiles.Count; i++)
181             {
182                 if (currentTiles[i] == 0)
183                     availablePositions.Add(i);
184             }
185 
186             if (availablePositions.Count == 0)
187                 throw new ArgumentException("No available playable tiles.");
188 
189             var random = new Random();
190             var randomTileIndex = random.Next(0, availablePositions.Count);
191 
192             return availablePositions[randomTileIndex];
193         }
```

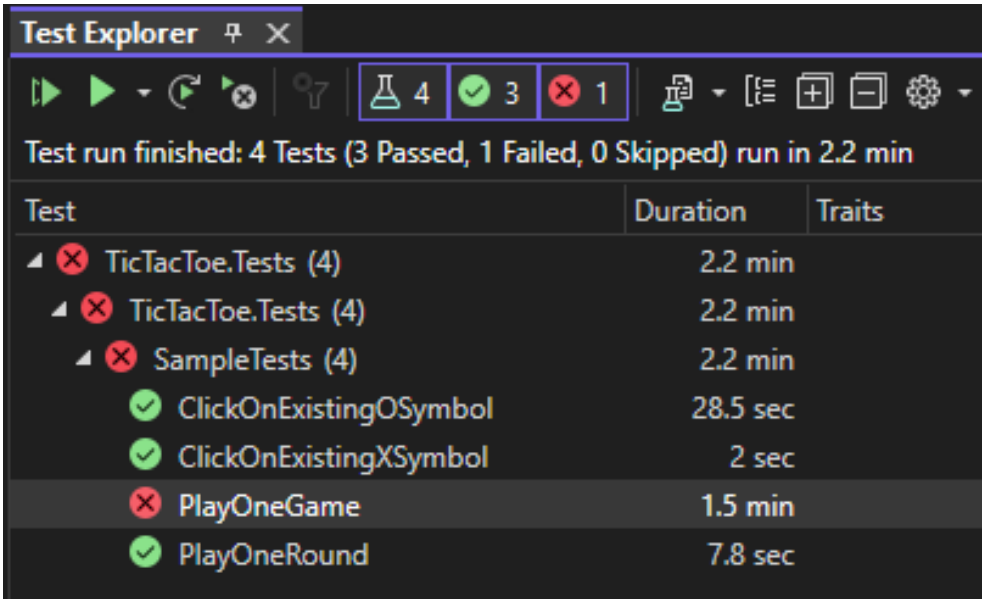
```
195         private string FindTilePosition(int tileNumber)
196         {
197             if (tileNumber > 8 || tileNumber < 0)
198                 throw new ArgumentException("Tic tac toe game does not contain more than three rows.");
199 
200             var rowNumber = string.Empty;
201 
202             if (tileNumber < 3)
203                 return "0" + tileNumber.ToString();
204 
205             else if (tileNumber > 2 && tileNumber < 6)
206             {
207                 rowNumber = "1";
208                 if (tileNumber == 3)
209                     return rowNumber + "0";
210 
211                 else if (tileNumber == 4)
212                     return rowNumber + "1";
213 
214                 else
215                     return rowNumber + "2";
216             }
217             else
218             {
219                 rowNumber = "2";
220                 if (tileNumber == 6)
221                     return rowNumber + "0";
222 
223                 else if (tileNumber == 4)
224                     return rowNumber + "1";
225 
226                 else
227                     return rowNumber + "2";
228             }
229         }
```

Τέλος, θα αναφέρουμε τις δοκιμές που επιλέξαμε να εκτελέσουμε για να διασφαλίσουμε την ποιότητα του παιχνιδιού. Καθώς αυτές που εκτελούνται επιτυχώς, όπως και αυτές που εκτελούνται ανεπιτυχώς. Η δοκιμή που εκτελείται ανεπιτυχώς, είναι η “PlayOneGame”. Πρόκειται για μία δοκιμή που σαν στόχο έχει να δοκιμάσει ότι ένας χρήστης μπορεί να παίξει έναν ολοκληρωμένο γύρο. Θα επεξηγήσουμε τους λόγους που αφήσαμε την δοκιμή αυτή να εμφανίζεται σαν αποτυχημένη στην σουίτα μας, για να αναδείξουμε και έναν από τους περιορισμούς του μηχανισμού που έχουμε υλοποιήσει. Οι υπόλοιπες δοκιμές είναι οι εξής, με τη σειρά που εμφανίζονται παρακάτω:

Η ”ClickOnExistingOSymbol”, όπου η δοκιμή αλληλοεπιδρά επιτυχώς με το παιχνίδι, τοποθετώντας ένα σύμβολο “X” σε ένα τυχαίο κουτί της τρίλιζας. Στη συνέχεια εντοπίζει το κουτί, όπου έχει επιλεγεί από το παιχνίδι, να προστεθεί το σύμβολο “O”, και αλληλοεπιδρά πάνω του. Με την αλληλοεπίδραση αυτή, εμφανίζεται το AllertPopup του παιχνιδιου, και είμαστε σε θέση να δούμε το μήνυμα “Try another box!”. Με αυτόν τον τρόπο δοκιμάζουμε ότι η εφαρμογή δεν εμφανίζει κάποια αστοχία, όταν ο χρήστης επιλέγει ένα κουτί που έχει επιλεχθεί σε προηγούμενο γύρο από το παιχνίδι.

Η δεύτερη δοκιμή είναι η “ClickOnExistingXSymbol”. Είναι παρόμοια με την προηγούμενη, με την διαφορά ότι δεν ψάχνουμε να εντοπίσουμε ένα σύμβολο που έχει προστεθεί από το παιχνίδι, αλλά εντοπίζουμε εάν το σύμβολο “X”, στο τυχαίο κουτί που έχει επιλέξει η δοκιμή εμφανίζεται και αλληλοεπιδράμε με αυτό. Στην συνέχεια ο έλεγχος που πραγματοποιούμε, είναι παρόμοιος με την προηγούμενη δοκιμή. Το AlertPopup πρέπει να εμφανιστεί και να εμπεριέχει το σωστό μήνυμα. Η τελευταία δοκιμή είναι η “PlayOneRound”. Σε αυτή η δοκιμή τοποθετεί ένα σύμβολο “X” σε ένα τυχαίο κουτί της τρίλιζας, και στην συνέχει ελέγχουμε εάν το παιχνίδι έχει τοποθετήσει ένα σύμβολο “O”.

5.2.1.19 Η Σουίτα Δοκιμών



5.2.1.20 Η κλάση SampleTests (Οι δοκιμές "ClickOnExistingXSymbol" και "ClickOnExistingOSymbol")

```
57          [Test]
58          public void ClickOnExistingXSymbol()
59          {
60              var errorMessage = new AlertPopup(_driver);
61              _gamePage.SetDefaultSizeWindow();
62              _gamePage.IsTileSymbolExist(this.FindTilePosition(0), emptyTile.ToString());
63              var currentXPlayableTile = this.FindRandomPlayableTile(gameTiles);
64              _gamePage.PlayOneRound(this.FindTilePosition(currentXPlayableTile), emptyTile.ToString(), xTile.ToString());
65              _gamePage.PlayOneRound(this.FindTilePosition(currentXPlayableTile), xTile.ToString());
66              Assert.AreEqual(errorMessage.Message, "Try another box!");
67          }
68
69          [Test]
70          public void ClickOnExistingOSymbol()
71          {
72              var errorMessage = new AlertPopup(_driver);
73              _gamePage.SetDefaultSizeWindow();
74              _gamePage.IsTileSymbolExist(this.FindTilePosition(0), emptyTile.ToString());
75              var currentXPlayableTile = this.FindRandomPlayableTile(gameTiles);
76              _gamePage.PlayOneRound(this.FindTilePosition(currentXPlayableTile), emptyTile.ToString(), xTile.ToString());
77              bool opponentHasPlayed = false;
78              var currentAvailableOppentTiles = this.FindAllAvailableGameTiles(gameTiles);
79              int existingOTile = -1;
80              foreach (var currentOpponentTile in currentAvailableOppentTiles)
81              {
82                  opponentHasPlayed = _gamePage.IsTileSymbolExist(this.FindTilePosition(currentOpponentTile), oTile.ToString());
83                  if (opponentHasPlayed)
84                  {
85                      gameTiles[currentOpponentTile] = oTile;
86                      existingOTile = currentOpponentTile;
87                      break;
88                  }
89              }
90
91              Assert.AreNotEqual(existingOTile, -1, "Cannot find openent tile.");
92              _gamePage.PlayOneRound(this.FindTilePosition(existingOTile), oTile.ToString());
93
94              Assert.AreEqual(errorMessage.Message, "Try another box!");
95          }
```

5.2.1.21 Η κλάση SampleTests (Η δοκιμή “ PlayOneRound”)

```
97          [Test]
98          public void PlayOneRound()
99          {
100             _gamePage.SetDefaultSizeWindow();
101             var currentXPlayableTile = this.FindRandomPlayableTile(gameTiles);
102             _gamePage.PlayOneRound(this.FindTilePosition(currentXPlayableTile), emptyTile.ToString(), xTile.ToString());
103             var currentAvailableOppentTiles = this.FindAllAvailableGameTiles(gameTiles);
104
105             bool opponentHasPlayed = false;
106             foreach (var currentOpponentTile in currentAvailableOppentTiles)
107             {
108                 opponentHasPlayed = _gamePage.IsTileSymbolExist(this.FindTilePosition(currentOpponentTile), oTile.ToString());
109                 if (opponentHasPlayed)
110                 {
111                     gameTiles[currentOpponentTile] = oTile;
112                     break;
113                 }
114             }
115
116             Assert.IsTrue(opponentHasPlayed, "Program cannot regognize oponent's symbol.");
117         }
```

5.2.1.22 Η κλάση SampleTests (Η δοκιμή “ PlayOneGame”)

```
119         [Test]
120         public void PlayOneGame()
121         {
122             var alertPopup = new AlertPopup(_driver);
123             _gamePage.SetDefaultSizeWindow();
124
125             int count = 1;
126             bool winMessage = false;
127             bool opponentHasPlayed = false;
128             do
129             {
130                 opponentHasPlayed = false;
131                 var currentXPlayableTile = this.FindRandomPlayableTile(gameTiles);
132                 gameTiles[currentXPlayableTile] = xTile;
133                 _gamePage.PlayOneRound(this.FindTilePosition(currentXPlayableTile), emptyTile.ToString(), xTile.ToString());
134                 Assert.IsTrue(_gamePage.IsTileSymbolExist(this.FindTilePosition(currentXPlayableTile), xTile.ToString()));
135                 var currentAvailableOppentTiles = this.FindAllAvailableGameTiles(gameTiles);
136                 foreach (var currentOpponentTile in currentAvailableOppentTiles)
137                 {
138                     opponentHasPlayed = _gamePage.IsTileSymbolExist(this.FindTilePosition(currentOpponentTile), oTile.ToString());
139                     if (opponentHasPlayed)
140                     {
141                         gameTiles[currentOpponentTile] = oTile;
142                         Debug.WriteLine("currentOpponentTile: " + currentOpponentTile);
143                         break;
144                     }
145                 }
146
147                 //If the foreach block code ended and the program did not find the oponent's game tile then assert and break the test with Failed status.
148                 Assert.True(opponentHasPlayed, "Program cannot regognize oponent's symbol.");
149                 //If the alert pop up is displayed, it means that the program falsy clicked on tile that contains X or O symbol.
150                 Assert.False(alertPopup.IsVisible, "the program falsy clicked on tile that contains X or O symbol");
151
152                 if (_gamePage.HasSomeOneWon)
153                     winMessage = true;
154
155             } while (winMessage == false && count < 9);
156
157             if (opponentHasPlayed == false && count == 9)
158                 Assert.True(opponentHasPlayed, "Program cannot regognize oponent's symbol after 9 times.");
159
160             Assert.IsTrue(winMessage, "Someone has won !!!");
161         }
```

Στο παρακάτω κεφάλαιο, θα προχωρήσουμε στην επεξήγηση των περιορισμών του μηχανισμού αναγνώρισης, απαντώντας και στο ερώτημα εάν μπορεί η υλοποίηση μας να θεωρηθεί χρήσιμη, σε μία πιο ευρεία χρήση της, για τους σκοπούς της διαφύλαξης της ποιότητας του λογισμικού, μίας εφαρμογής ιστού.

6. Αποτελέσματα

Αναφορικά με τα ευρήματα της εργασίας, σε αυτό το κεφάλαιο θα παρουσιάσουμε κάποιους από τους περιορισμούς και τα προβλήματα που ανέκυψαν κατά την υλοποίηση. Καθώς, θα αναφέρουμε και θα υποστηρίξουμε κάποιες από τις λύσεις, που μπορούν να χρησιμοποιηθούν από τους μηχανικούς, κατά τη χρήση της αναγνώρισης εικόνας.

Ο πρώτος περιορισμός που εντοπίσαμε, έχει να κάνει με τις διαστάσεις που έχουμε επιλέξει να εξαγάγουμε το πρότυπο, και στη χρήση του σε διαφορετικές διαστάσεις. Πρόκειται για έναν περιορισμό της αποκριτικότητας της εφαρμογής ιστού. Οι σύγχρονες εφαρμογές ιστού, είναι υλοποιημένες με τέτοιο τρόπο, ώστε να μπορούν να εφαρμόζουν σε διαφορετικές διαστάσεις, ακόμα και συσκευές. Υπάρχουν πολλές εφαρμογές, με μία κοινή βάση κώδικα, οι οποίες είναι υλοποιημένες με χρήση συγκεκριμένων τεχνολογιών, για να τρέχουν τόσο σε προσωπικούς υπολογιστές, αλλά και σε κινητά τηλέφωνα.

Ο συγκεκριμένος περιορισμός, μπορεί να εντοπισθεί ακόμα και στην χρήση της αναγνώρισης εικόνας, με την χρήση ενός περιηγητή σε έναν προσωπικού υπολογιστή. Καθώς, εάν έχουμε εξαγάγει το πρότυπο από ένα στιγμιότυπο της εφαρμογής, σε διαστάσεις 500x500, και προσπαθήσουμε να κάνουμε χρήση του μηχανισμού αναγνώρισης σε μεγαλύτερες ή μικρότερες διαστάσεις παραθύρου, η αναγνώριση δεν θα επιτευχθεί. Αυτό συνάδει, με τον τρόπο που η παρεχόμενη από την βιβλιοθήκη OpenCvSharp στατική μέθοδος αντιστοίχισης εικόνων MatchTemplate, είναι υλοποιημένη. Ουσιαστικά, μπορούμε να φανταστούμε την μέθοδο, σαν να ξεκινάει να σέρνει το πρότυπο, από ένα σημείο του στιγμιότυπου, προσπαθώντας να επιτύχει την αντιστοίχιση. Αυτό έχει σαν αποτέλεσμα, σε μεγαλύτερες διαστάσεις, το πρότυπο να είναι μικρότερο του αντικειμένου εκείνη την χρονική στιγμή. Ενώ σε μικρότερες, το πρότυπο θα είναι μεγαλύτερο.

Είμαστε υποχρεωμένοι να αναφέρουμε αυτό τον περιορισμό, αλλά επίσης δεν τον θεωρούμε απαγορευτικό. Καθώς, μπορούμε να ορίσουμε κάποιες σταθερές διαστάσεις στο πρόγραμμα περιήγησης που ελέγχετε από το Selenium, κατά την εκκίνηση του περιηγητή, όσο και αργότερα κατά την εκτέλεση της δοκιμής. Αυτές οι παρεχόμενες μέθοδοι του Selenium, μας επιτρέπουν να ορίσουμε της διαστάσεις όπου θα εξαχθούν τα πρότυπα, αλλά και θα εκτελούνται οι δοκιμές. Εάν όμως, ο μηχανικός χρειαστεί να εκτελέσει δοκιμές σε διαφορετικές διαστάσεις, είναι απαραίτητο να εξαγάγει, έναν σημαντικά μεγαλύτερο αριθμό προτύπων, και κατά την εκτέλεση των δοκιμών να κάνει χρήση κάποιων δικών του ορισμάτων, ώστε να επιλέγει το αντίστοιχο πρότυπο, με βάση τις διαστάσεις.

Ένας δεύτερος περιορισμός, έχει να κάνει με την αντιστοίχιση του προτύπου, σε ένα στιγμιότυπο που εμπεριέχει πολλές ίδιες εικόνες. Εάν έχουμε κόψει τις εικόνες που θα χρησιμοποιήσουμε σαν locators, με μία σχετική ακρίβεια στα όρια του κάθε κουτιού της τρίλιζας, αυτό δημιουργεί πρόβλημα στην συνάρτηση αναγνώρισης. Ενδέχεται η συνάρτηση αναγνώρισης, να βρίσκει παρόμοια σύμβολα, τοποθετημένα από προηγούμενους γύρους.

Για παράδειγμα, εάν έχει ολοκληρωθεί ένας πρώτος γύρος του παιχνιδιού, και υπάρχει ένα σύμβολο "X" και ένα σύμβολο "O", η αναγνώριση μπορεί να χαρακτηριστεί ως επιτυχής. Εάν όμως, η δοκιμή προχωρήσει σε ένα δεύτερο γύρο, και τοποθετήσει ένα δεύτερο σύμβολο "X", καθώς και το παιχνίδι εμφανίσει ένα δεύτερο σύμβολο "O", τότε ενδέχεται στην προσπάθεια του εντοπισμού του δεύτερου συμβόλου, είτε πρόκειται για το σύμβολο "X", είτε για το σύμβολο "O", ο μηχανισμός εντοπισμού να αναγνωρίσει το προηγούμενο σύμβολο και να θεωρήσει ότι η αναγνώριση είναι επιτυχής. Έχοντας αναγνωρίσει εσφαλμένα το σύμβολο του πρώτου γύρου, αντί του δευτέρου, μπορεί να μπερδέψει την δοκιμή σε δύο στάδια.

Στις δοκιμές, μετά από κάποια ενέργεια στο πρόγραμμα περιήγησης, χρησιμοποιούνται κάποιοι μηχανισμοί, που στόχο έχουν την επιβράδυνση της ταχύτητας του προγράμματος, μέχρις ότου να εμφανιστεί κάποια ζητούμενη από την δοκιμή αλλαγή στο DOM. Αυτοί οι μηχανισμοί εν συντομία ονομάζονται "Waits". Μπορούμε να χρησιμοποιήσουμε τριών ειδών "Waits", όπως τα "Implicit", "Explicit" και "Fluent". Συνήθως αυτοί οι μηχανισμοί, προσπαθούν να εντοπίσουν εάν ένα αντικείμενο εμφανίζεται, εξαφανίζεται, μπορεί να επιλεγεί και να χρησιμοποιηθεί πάνω του η μέθοδος "Click", εμπεριέχει κάποιο κείμενο ή χαρακτηριστικό.

Το Selenium εμπεριείχε αυτούς τους μηχανισμούς μέχρι πρότινος, σε παλαιότερες εκδόσεις του, αλλά πλέον έχουν αφαιρεθεί. Για να κάνουμε χρήση αυτών, πρέπει να εγκαταστήσουμε της βιβλιοθήκη "SeleniumExtras". Θα βρούμε τους εν λόγω μηχανισμούς στο namespace "SeleniumExtras.WaitHelpers".

Όπως είδαμε και στο παραπάνω κεφάλαιο, έχουμε δημιουργήσει και εμείς έναν τέτοιο μηχανισμό, την μέθοδο "UntiGraphicalElementIsVisible". Που σε σχέση με μεθόδους του Selenium, δεν κάνει κάτι διαφορετικό από το να ψάχνει ένα αντικείμενο για έναν καθορισμένο χρονικό διάστημα, που του ορίζουμε εμείς. Εάν το αντικείμενο εντοπιστεί μέσα στο καθορισμένο χρονικό διάστημα, τότε το πρόγραμμα συνεχίζει κανονικά την εκτέλεση του. Εάν δεν καταφέρει να εντοπίσει το αντικείμενο, στην συγκεκριμένη περίπτωση το πρότυπο εικόνας, τότε εμφανίζεται το Exception

τύπου “WebDriverTimeoutException”. Εφόσον λοιπόν η δοκιμή τοποθετήσει ένα δεύτερο σύμβολο “X”, και γίνει η κλήση της μεθόδου “UntiGraphicalElementIsVisible”, θα εντοπιστεί το προϋπάρχον σύμβολο, μην καταφέρνοντας η μέθοδος να χρησιμοποιηθεί με την ορθή χρήση της. Μετά από κάθε τοποθέτηση του συμβόλου “X”, η δοκιμή προσπαθεί να εντοπίσει το σύμβολο “O”, που έχει προστεθεί από το παιχνίδι.

Στην δεύτερη περίπτωση, εάν το παιχνίδι έχει κάποια δυσλειτουργία, και το δεύτερο σύμβολο “O” δεν εμφανιστεί, η μέθοδος θα εντοπίσει το σύμβολο του πρώτου γύρου. Τότε η δοκιμή θα οδηγηθεί σε λάθος αποτελέσματα. Εφόσον, θα αναγνωρίζει εσφαλμένα ότι το σύμβολο έχει τοποθετηθεί σε μία υποτιθέμενη θέση. Αυτό συμβαίνει, παρότι δίνονται στην συνάρτηση αναγνώρισης συγκεκριμένες τιμές του σχήματος και του ιστογράμματος του προτύπου, που έχουμε εξάγει. Ακόμα και αν αυτές οι εικόνες εμπεριέχουν σε διαφορετικά σημεία τις γραμμές του κουτιού, που έχουμε τοποθετήσει ηθελημένα, σε μία προσπάθεια να προσδώσουμε μοναδικότητα στο κάθε πρότυπο.

Από ότι συμπεραίνουμε, η μόνη λύση σε αυτό το πρόβλημα, με την συγκεκριμένη υλοποίηση, είναι η προσπάθεια να γίνει όσο πιο δυνατόν διαφορετικό το κάθε πρότυπο που έχει εξαχθεί, για το εκάστοτε κουτί. Αυτό μπορεί να πραγματοποιηθεί αλλάζοντας το σχήμα ή το ιστόγραμμα του κάθε συμβόλου, με βάση σε πιο κουτί της τρίλιζας εμπεριέχεται, ή και τοποθετώντας κατά την εξαγωγή του προτύπου, άλλα μοναδικά αντικείμενα, που μπορεί να βρίσκονται κοντά του. Στόχος είναι, το κάθε κουτί από τα εννέα της τρίλιζας, να έχει τρία μοναδικά πρότυπα, σε σχέση με τα πρότυπα των υπόλοιπων κουτιών. Ένα με κενό κουτί, ένα με το σύμβολο “X” και ένα με το σύμβολο “O”. Και τα τρία πρότυπα πρέπει να εμπεριέχουν και κάποιο άλλο αντικείμενο. Αυτό δίνει την μοναδικότητα που χρειαζόμαστε, αλλά θα πρέπει να συμβεί σχεδόν για κάθε ένα κουτί της τρίλιζας.

Είναι γνωστή η πρακτική, η τοποθέτηση από τους μηχανικούς που έχουν υλοποιήσει μία εφαρμογή ιστού, σε συνεννόηση με τους μηχανικούς διασφάλισης λογισμικού, να προσθέτουν στα διάφορα αντικείμενα του DOM, κάποια επιπλέον χαρακτηριστικά πάνω στα αντικείμενα. Αυτό έχει σαν σκοπό, την διευκόλυνση της πραγματοποίησης αυτοματοποιημένων δοκιμών. Κάτι τέτοιο δεν μπορεί να εφαρμοστεί στην συγκεκριμένη εφαρμογή ιστού, καθώς τα αντικείμενα του DOM παραμένουν κριμένα με την χρήση της τεχνολογίας Canva. Επιπροσθέτως, όποια παρέμβαση στην εμφάνιση της εφαρμογής για τους σκοπούς της αυτοματοποίησης, είναι αρκετά

δύσκολο να αποφασιστεί και να υλοποιηθεί. Εξάλλου οι μηχανικοί καλούνται να αυτοματοποιήσουν υπάρχουσες εφαρμογές και όχι να θέσουν, όρους που επηρεάζουν την εμφάνιση της εφαρμογής, με σκοπό την επιτευξη της αυτοματοποίησης της.

Εν κατακλείδι, παρά τους περιορισμούς που αναφέρθηκαν στο κεφάλαιο, θεωρούμε ότι παρέχεται μία λύση στο πρόβλημα των κρυμμένων αντικειμένων. Με τον συνδυασμό της βιβλιοθήκης OpenCvSharp και του Selenium, ενισχύονται σημαντικά οι δυνατότητες ενός property-based GUI εργαλείου, και δίνεται μία λύση στον εκάστοτε μηχανικό για την χρήση του, έστω και κάτω από κάποιους περιορισμούς.

7. Βιβλιογραφία

Trade-off between automated and manual software testing, Ossi Taipale, Jussi Kasurinen, Katja Karhu, Kari Smolander

Visual GUI Testing: Automating High-level Software Testing in Industrial Practice. Chalmers University of Technology 2015, Emil Alegroth

Visual GUI Testing: Automating High-Level Software Testing in Industrial Practice

The C# Yellow book, Rob Miles

https://learn.microsoft.com/en-us/archive/msdn-magazine/2014/august/unity-developing-your-first-game-with-unity-and-csharp

https://learn.microsoft.com/en-us/dotnet/standard/clr

https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet-framework

https://learn.microsoft.com/en-us/nuget/what-is-nuget

https://www.toolsqa.com/selenium-webdriver/selenium-webdriver-architecture

https://en.wikipedia.org/wiki/Selenium_(software)

Online documentation of C# language

https://en.wikipedia.org/wiki/HTML

https://www.w3schools.com/html/html_intro.asp

https://www.w3schools.com/css/

https://en.wikipedia.org/wiki/CSS

https://getbootstrap.com/

https://en.wikipedia.org/wiki/JavaScript

https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial

https://nodejs.org/en/about/

https://en.wikipedia.org/wiki/Node.js

https://en.wikipedia.org/wiki/Npm_(software)

https://docs.npmjs.com/creating-a-package-json-file

https://fonts.google.com/about

https://pages.github.com/

https://www.w3schools.com/html/html_iframe.asp

https://docs.opencv.org/4.x/d4/dc6/tutorial_py_template_matching.html
https://docs.opencv.org/4.x/de/db2/tutorial_py_table_of_contents_histograms.html
https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html