ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΕΛΕΤΗ ΚΑΙ ΑΝΑΠΤΥΞΗ ΕΡΓΑΛΕΙΩΝ ΣΥΝΕΧΟΥΣ ΕΝΣΩΜΑΤΩΣΗΣ (CONTINUOUS INTEGRATION) ΣΕ ΣΥΓΧΡΟΝΑ ΕΡΓΑ ΛΟΓΙΣΜΙΚΟΥ

Διπλωματική Εργασία
του
Λουκίδη Γρηγόριου

Θεσσαλονίκη, Ιούνιος 2019

ΜΕΛΕΤΗ ΚΑΙ ΑΝΑΠΤΥΞΗ ΕΡΓΑΛΕΙΩΝ ΣΥΝΕΧΟΥΣ ΕΝΣΩΜΑΤΩΣΗΣ (CONTINUOUS INTEGRATION) ΣΕ ΣΥΓΧΡΟΝΑ ΕΡΓΑ ΛΟΓΙΣΜΙΚΟΥ

Γρηγόριος Λουκίδης

Πτυχίο Εφαρμοσμένης Πληροφορικής, Πανεπιστήμιο Μακεδονίας, 2016

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων Καθηγητής
Αλέξανδρος Χατζηγεωργίου

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ηη/μμ/εεεε

Αλέξανδρος Χατζηγεωργίου, Καθηγητής	Ελευθέριος Μαμάτας	Ηλίας Σακελλαρίου
.....	.....	.....

Γρηγόριος Λουκίδης

.....

Περίληψη

Η έννοια της συνεχούς ενσωμάτωσης (Continuous Integration) αποτελεί θεμελιώδη διαδικασία στην ανάπτυξη σύγχρονων έργων λογισμικού. Βασικός πυλώνας της θεωρείται η αξιολόγηση και βελτιστοποίηση του κώδικα και της παραμετροποίησης του έργου μέσα από ρητά καθορισμένες σειρές ενεργειών ώστε το λογισμικό να είναι λειτουργικά άρτιο και όσο το δυνατόν βέλτιστο.

Σε μια βασική ροή ανάπτυξης και συντήρησης μίας εφαρμογής βασικό ρόλο παίζει το merging (ενσωμάτωση). Κάθε προγραμματιστής όταν προσθέσει μια νέα επέκταση, διόρθωση ή αλλαγή στο λογισμικό, αρχικά εργάζεται τοπικά στον υπολογιστή του και εφόσον η εργασία του είναι αποδεκτή από μία σειρά κανόνων ενσωματώνεται στην κυρίως εφαρμογή ως νέα έκδοση (new release).

Σκοπός της παρούσας διπλωματικής εργασίας είναι να αναλύσει τα βήματα ώστε οι νέες προσθήκες να είναι ποιοτικά και λειτουργικά αποδεκτές, καθώς επίσης και τη διαδικασία της ανάπτυξης (deploy/release) των αλλαγών στο περιβάλλον το οποίο χρησιμοποιεί ο τελικός χρήστης. Συνεπώς αποτελείται από μια εφαρμογή στην οποία έχουν πρόσβαση χρήστες μέσω διαδικτύου και αλληλεπιδρούν και από μία τοπική εφαρμογή η οποία “βελτιώνει και ελέγχει” την πρώτη. Αυτό επιτυγχάνεται με την εκτέλεση tests, αναλύσεων ποιότητας του κώδικα, build (χτίσιμο του project τοπικά), επικοινωνία με τη βάση, διατήρηση εκδόσεων του κώδικα μέσω αντίστοιχων εργαλείων καθώς και ανάπτυξης του σε server. Τέλος στον server ξαναχτίζεται το project και είναι έτοιμο για λειτουργία.

Λέξεις κλειδιά: συνεχής ενσωμάτωση, build, release, server, λογισμικό, κώδικας, έκδοση, back-end, front-end

Abstract

The concept of Continuous Integration is a fundamental process in the development of modern software projects. Its main pillar is the evaluation and optimization of the project's code and customization through explicitly defined sets of actions so that the software is as functional and as optimally as possible.

In a basic flow of development and maintenance of an application, merging (embedding) plays a key role. Each developer when adding a new extension, patch or change to the software initially works locally on his computer, and if his work is accepted by a set of rules, it is incorporated into the main application as a new release.

The purpose of this diploma thesis is to analyze the steps for the new additions to be qualitatively and operationally acceptable, as well as the process of deploying / releasing the changes in the environment that the end user uses. It therefore consists of an application accessible to users via the Internet and interacting with a local application that "improves and controls" the first one. This is accomplished by running tests, analyzing the quality of the code, building (locally building the project), communicating with the database, maintaining code versions through corresponding tools, and deploying it on a server. Finally, the project is rebuilt in live environment and it gets ready for operation.

Keywords: continuous integration, build, release, server, software, code, version, back-end, front-end

Πρόλογος – Ευχαριστίες

Κάπου εδώ θα ήθελα να ευχαριστήσω τον κ. Χατζηγεωργίου Αλέξανδρο για την άριστη συνεργασία που είχαμε τον τελευταίο χρόνο και για την καθοδήγησή του. Ιδιαίτερα θετικό για την εκπόνηση της διπλωματικής εργασίας ήταν το γεγονός ότι ήταν ανοιχτός σε προτάσεις μου χωρίς εν τέλει να υπάρξει απόκλιση από τον στόχο. Τον ευχαριστώ ιδιαίτερα γιατί είναι ένας εξαιρετικός άνθρωπος και μέσα από τις διαλέξεις του σε προπτυχιακό επίπεδο έλαβα σημαντικά κίνητρα για να ασχοληθώ επαγγελματικά με τον προγραμματισμό και να συνεχίσω τις σπουδές μου σε μεταπτυχιακό επίπεδο.

Θα ήθελα επίσης να πω ένα μεγάλο ευχαριστώ στη Χρυσούλα για τη βοήθεια της, και την υπομονή της τα τελευταία 3 χρόνια. Ευχαριστώ όλους του φίλους, συναδέλφους και κυρίως την οικογένειά μου για τη στήριξη και την αναγνώριση τον κόπων μου.

Περιεχόμενα

Κατάλογος Εικόνων

1 Εισαγωγή

1.1 Πρόβλημα – Σημαντικότητα του θέματος

Σε σύγχρονα έργα λογισμικού γίνεται λόγος για την έννοια της συνεχούς ενσωμάτωσης (Continuous Integration). Για να φτάσει ένα έργο λογισμικού στην τελική του μορφή και να είναι έτοιμο προς λειτουργία στον τελικό χρήστη απαιτείται μια σημαντική σειρά ενεργειών. Ποιες είναι ωστόσο αυτές οι ενέργειες, ποια η σειρά τους, πως καθορίζεται η αναγκαιότητά τους και η σημαντικότητά κάθε μιας ξεχωριστά σε ένα κύκλο συνεχούς ενσωμάτωσης; Πως μπορούν να εφαρμοστούν σε ένα έργο λογισμικού; Ποιες τεχνολογίες μπορούν να συνδυαστούν για να επιτευχθεί η ανάπτυξη ενός εργαλείου το οποίο θα έχει ως σκοπό της περάτωση κύκλων συνεχούς ενσωμάτωσης.

1.2 Σκοπός – Στόχοι

Σκοπός της παρούσας εργασίας είναι αρχικά να μελετήσει τις διαδικασίες και τα εργαλεία που μπορούν να συνεισφέρουν στην επιτυχημένη και ποιοτική ανάπτυξη ενός έργου λογισμικού, και έπειτα να αναπτύξει ένα έργο λογισμικού και να εφαρμόσει πάνω σε αυτό τις διαδικασίες και τις τεχνικές συνεχούς ολοκλήρωσης μέσω ενός δεύτερου έργου λογισμικού. Σημαντικός στόχος επίσης είναι να καταγράψει τις τεχνολογίες που μπορούν να συντελέσουν στην ανάπτυξη αυτών των έργων καθώς και τα σκοτεινά σημεία τους. Αυτό θα επιτευχθεί με την ανάπτυξη δύο έργων λογισμικού. Το εργαλείο συνεχούς ενσωμάτωσης (εφαρμογή 2) θα "τρέχει" συνεχείς κύκλους ενσωμάτωσης πάνω στην εφαρμογή 1, ένα project που συνδυάζει τεχνολογίες διαδικτύου και έχει business υπόβαθρο.

1.3 Διάρθρωση της μελέτης

Η διάρθρωση της μελέτης έχει ως εξής. Ανάλυση του θεωρητικού και τεχνολογικού υπόβαθρου συναντάμε στο κεφάλαιο 2. Εκεί επίσης παρουσιάζεται η επιλογή του εκάστοτε εργαλείου, τεχνολογίας ή πλατφόρμας. Το Κεφάλαιο 3 συζητά τη διαδικασία ανάπτυξης των δύο εφαρμογών, τις δυσκολίες, τις επαναπροσεγγίσεις και τις αλλαγές ροής εργασίας που προέκυψαν. Στο Κεφάλαιο 4 αναπτύσσουμε μια σειρά αποτελεσμάτων, συμπερασμάτων και πληροφοριών που πηγάζουν από την ανάπτυξη των δύο εφαρμογών και μια σειρά κύκλων συνεχούς ενσωμάτωσης μεταξύ τους.

2 Τεχνολογικό – Θεωρητικό Υπόβαθρο

2.1 Τι είναι το Continuous Integration

Το Continuous Integration (Συνεχής Ενσωμάτωση) αποτελεί μια σύγχρονη πρακτική ανάπτυξης έργων λογισμικού, η οποία απαιτεί από τους προγραμματιστές να ενσωματώνουν τις αλλαγές στον κώδικα τους σε ένα κοινό branch (κλαδί) πολλές φορές μέσα σε μία μέρα. [1] Κάθε κύκλος συνεχούς ενσωμάτωσης βασίζεται σε ελέγχους και build , οι οποίοι μπορούν σχετικά "νωρίς" να εντοπίσουν τα όποια λάθη και αστοχίες και με τα κατάλληλα μηνύματα να βοηθήσουν τον προγραμματιστή να τα διορθώσει. Ο Grady Booch πρώτος πρότεινε τον όρο Continuous Integration, το 1991, ως ένα υποσύνολο του Extreme Programming (XP), αν και ο ίδιος δεν υπήρξε συνήγορος της ενσωμάτωσης πολλές φορές μέσα στην ίδια μέρα.

Ο ευρύτερος σκοπός της συνεχούς ενσωμάτωσης είναι η πρόληψη αλλά και η εντόπιση σφαλμάτων σε όσο το δυνατόν πιο πρώιμο στάδιο μπορεί να συμβεί. Ο Martin Fowler αναφέρει "Η συνεχής ενσωμάτωση δεν ξεφορτώνεται τα σφάλματα, ωστόσο καθιστά δραματικά ευκολότερο τον εντοπισμό τους και την απαλοιφή τους." Όσο περισσότερα βήματα περιέχει ένας κύκλος συνεχούς ενσωμάτωσης, τόσο ποιοτικότερο θα προκύψει το έργο λογισμικού που θα εξέλθει της διαδικασίας. Αξίζει να σημειωθεί ό,τι η συνεχής ενσωμάτωση αποτελεί αναπόσπαστο κομμάτι στην ανάπτυξη σύγχρονων έργων λογισμικού. [1]

2.2 Πλεονεκτήματα της Συνεχούς Ενσωμάτωσης

Μείωση του χρόνου αποσφαλμάτωσης (debugging) και αφιέρωση περισσότερου στην ανάπτυξη νέων χαρακτηριστικών. [2]

Είναι χαρακτηριστικό ότι η συνεχής ενσωμάτωση εντοπίζει την πλειονότητα των σφαλμάτων, αλλά το σημαντικότερο είναι ότι κατευθύνει τον προγραμματιστή μέσω μηνυμάτων, αρχείων καταγραφής και αναφορών σε γρήγορο εντοπισμό τους διόρθωσή τους. Για παράδειγμα σε ένα προηγμένο σύστημα συνεχούς ενσωμάτωσης, ο χρήστης λαμβάνει σε μορφή μηνύματος, τη γραμμή και τη στήλη στον κώδικα του όπου εντοπίστηκα το σφάλμα, καθώς επίσης τον τύπο του σφάλματος αλλά και πιθανές προτάσεις επίλυσής του. Έτσι μειώνεται ραγδαία ο χρόνος εντοπισμού και διόρθωσης και αξιοποιείται στην ανάπτυξη κώδικα.

Αυτοματοποίηση πολύπλοκων διαδικασιών

Ένας κύκλος συνεχούς ενσωμάτωσης δίνει τη δυνατότητα σύμπτυξης και αυτοματοποίησης σύνθετων διαδικασιών και ελέγχων οι οποίες μπορούν να πραγματοποιηθούν "χειροκίνητα" μεν, αλλά με μεγαλύτερο ποσοστό λάθους και αναπόφευκτα υψηλότερο χρόνο εκτέλεσης.[2] Σε ένα σύστημα συνεχούς ολοκλήρωσης ο προγραμματιστής μπορεί να κάνει build (μεταγλώττιση και σύνδεση του κώδικα) με το πάτημα ενός κουμπιού, ωστόσο αυτό μπορεί να επιτευχθεί και με μια σειρά εντολών σε γραμμή εντολών, ή με άνοιγμα του περιβάλλοντος ανάπτυξης του κώδικα και πάτημα ενός ή περισσότερων κουμπιών από εκεί.[15] Συνεπώς ο προγραμματιστής κερδίζει πολύτιμο χρόνο και έχει έτοιμη όλη τη διαδικασία, χωρίς να χρειάζεται να εισέλθει σε λεπτομέρειες εκτέλεσης.

Συνολικός έλεγχος όλων των σταδίων ενσωμάτωσης.

Η εκτέλεση των σταδίων (build, έλεγχοι, ενσωμάτωση) γίνεται σε μια σειρά βημάτων η οποία είναι αυστηρά καθορισμένοι. Ως αποτέλεσμα των παραπάνω, ο προγραμματιστής μέσω των μηνυμάτων που καταγράφονται στην οθόνη του, γνωρίζει σε ποιοι στάδιο του κύκλου ενσωμάτωσης βρίσκεται ανά πάσα στιγμή, ποια έχουν προηγηθεί και ποια έπονται.

Δυνατότητα εργασία πολλών προγραμματιστών παράλληλα και πολλαπλών ενσωματώσεων σε μικρό χρονικό διάστημα.

Θεμελιώδες χαρακτηριστικό της συνεχούς ενσωμάτωσης αποτελούν τα συστήματα διαχείρισης εκδόσεων λογισμικού, τα οποία ως σκοπό έχουν την ενσωμάτωση του κώδικα, τη διατήρηση ιστορικού εκδόσεων του κώδικα αλλά και την αποφυγή σύγχυσης μεταξύ των προγραμματιστών που εργάζονται παράλληλα στο ίδιο κομμάτι κώδικα. Με τη χρήση αυτών των συστημάτων δίνεται η δυνατότητα σε πολλούς προγραμματιστές να εργάζονται παράλληλα ακόμη και στα ίδια αρχεία.[15]

2.3 Μειονεκτήματα της Συνεχούς Ενσωμάτωσης

Αρκετές δυσκολίες στο στήσιμο μιας εφαρμογής σε σύστημα συνεχούς ενσωμάτωσης.
Παρά την ανάπτυξη και βελτίωση των συστημάτων CI, είναι ακόμη και σήμερα αρκετά πολύπλοκο για κάποιον χωρίς εμπειρία να στήσει την παραμετροποίηση και τους ελέγχους ώστε να εκτελείται συνεχής ενσωμάτωση σε μια εφαρμογή. Συνήθως απαιτούνται αρκετές αποτυχημένες προσπάθειες ώστε κάποιος χρήστης να φτάσει σε ένα σημείο σταθερότητας και αυτοματοποίησης. [2]

Κόστος υλικού και ασφάλεια.
Είναι σχεδόν επιτακτικό για όλα τα δημοφιλή συστήματα συνεχούς ενσωμάτωσης να εγκατασταθούν και να παραμετροποιηθούν σε ξεχωριστό server, με ισχυρά χαρακτηριστικά (μεγάλη μνήμη, δυνατοί επεξεργαστές κλπ), καθώς και επιτακτική ανάγκη ο server αυτός να επικοινωνεί με όλους τους τοπικούς υπολογιστές των προγραμματιστών, χωρίς ωστόσο να είναι ευάλωτος σε εξωτερικές κλήσεις, καθαρά για λόγους ασφαλείας.

2.4 Δημοφιλή Συστήματα Συνεχούς Ενσωμάτωσης

Jenkins CI
Το Jenkins, πρώην Hudson, είναι ένα εργαλείο συνεχούς ενσωμάτωσης που χρησιμοποιείται ευρέως και θεωρείται από πολλούς ως το de facto πρότυπο για το CI. Το Jenkins τρέχει σε Java, με δυνατότητα χρήσης σε Windows ή Linux. Λόγω του μεγάλου εύρους πρόσθετων (plug-in) που διαθέτει, το Jenkins έχει την ικανότητα να προσαρμόζεται σε projects διάφορων μεγεθών και αρκετά σύνθετων τεχνολογιών. Πολύ σημαντικά χαρακτηριστικά του είναι η δωρεάν χρήση του, καθώς και η πολύ μεγάλη υποστήριξη που παρέχεται στον εκάστοτε χρήστη μέσω της κοινότητας του στο διαδίκτυο η οποία είναι ιδιαίτερα αναπτυγμένη. Βασικό του μειονέκτημα αποτελεί το ξεπερασμένο User Interface του και οι δυσκολίες που συναντά κάποιος σχετικά άπειρος στην εκμάθηση των λειτουργιών του.[5] Για τις ανάγκες της συγκεκριμένης εργασίας επιλέχψθηκε το Jenkins καθώς είναι δωρεάν και με καλή υποστήριξη στο διαδίκτυο.

TeamCity
Αποτελεί ένα εργαλείο υψηλής γραφικής διεπαφής, ανεπτυγμένο από τη JetBrains. Η ευκολία χρήσης του, το καθιστά αρκετά πιο βατό για κάποιον σχετικά άπειρο στην έννοια της συνεχούς ενσωμάτωσης. Το TeamCity εκτελείται σε περιβάλλον Java,

συνήθως σε διακομιστή Apache Tomcat, αν και μπορεί να εγκατασταθεί είτε σε Windows και Linux servers. Επίσης προσφέρει ισότιμη υποστήριξη για έργα .NET και ανοιχτού λογισμικού, ενσωματώνοντας τόσο το Visual Studio όσο και άλλα IDE όπως το Eclipse.[5] Πολλά Jenkins projects έχουν μετατραπεί σε Team City projects λόγω της όμορφης διασύνδεσης του και της ασφαλής ρύθμισης παραμέτρων. Βασικό του μειονέκτημα αποτελούν οι περιορισμένοι πόροι της δωρεάν έκδοσής του. Για την υπερκάλυψη αυτών των δυνατοτήτων, απαιτείται η αγορά της εμπορικής έκδοσης.

2.5 Συστήματα στατικής ανάλυσης κώδικα

SonarQube

Το SonarQube (πρώην Sonar) είναι μια πλατφόρμα ανοικτού κώδικα που αναπτύχθηκε από την SonarSource για τη συνεχή επιθεώρηση της ποιότητας του κώδικα για την πραγματοποίηση αυτόματων αναθεωρήσεων με στατική ανάλυση του κώδικα για την ανίχνευση σφαλμάτων, “μυρωδιών” και ευπάθειας ασφαλείας σε 20+ γλώσσες προγραμματισμού.[13] Το SonarQube προσφέρει reports για πολλά πεδία όπως : πρότυπα κωδικοποίησης, δοκιμές μονάδων, κάλυψη κώδικα, πολυπλοκότητα κώδικα, σχόλια, σφάλματα και ευπάθειες ασφαλείας.

Το SonarQube μπορεί να καταγράψει το ιστορικό μετρήσεων και παρέχει γραφήματα εξέλιξης. Το SonarQube παρέχει πλήρως αυτοματοποιημένη ανάλυση και ολοκλήρωση με τα εργαλεία Maven, Ant, Gradle, MSBuild και συνεχή ενσωμάτωση (Atlassian Bamboo, Jenkins, Hudson κ.λπ.). Επιλέχθηκε καθώς παρέχει πολύ γρήγορη και πλήρη υλοποίηση για .NET projects.

RIPS Technologies

Το RIPS είναι η μόνη λύση ανάλυσης κώδικα που εκτελεί ανάλυση ασφαλείας συγκεκριμένης γλώσσας. Ανιχνεύει σύνθετες ευπάθειες ασφαλείας που είναι ενσωματωμένες στον πηγαίο κώδικα που δεν μπορούν να βρουν αρκετά άλλα εργαλεία.

Υποστηρίζει διάφορα frameworks, λειτουργίες συνεχούς ενσωμάτωσης, βιομηχανικά πρότυπα και μπορεί να αναπτυχθεί ως αυτοματοποιημένο λογισμικό ή ως λογισμικό-ως-υπηρεσία. Με την υψηλή ακρίβειά του και χωρίς ιδιαίτερες αποκλείσεις, το RIPS είναι η ιδανική επιλογή για την ανάλυση εφαρμογών Java και PHP. [4]

2.6 Συστήματα διαχείρισης εκδόσεων λογισμικού

SVN

Το SVN δημιουργήθηκε ως εναλλακτική λύση στο CVS που θα διορθώνει κάποια σφάλματα στο σύστημα CVS διατηρώντας παράλληλα υψηλή συμβατότητα με αυτό.
Όπως και το CVS, το SVN είναι ελεύθερο και ανοικτό, με τη διαφορά ότι διανέμεται υπό την άδεια του Apache σε αντίθεση με το GNU.

Πολλοί προγραμματιστές έχουν μεταπηδήσει στο SVN, καθώς είναι μια νεότερη τεχνολογία που υποστηρίζει τα καλύτερα χαρακτηριστικά του CVS και διαρκώς βελτιώνεται. Απευθύνεται κυρίως σε ομάδες προγραμματιστών οι οποίοι εργάζονται μαζί και έχουν όλοι πρόσβαση στον SVN server. Το βασικό του μειονέκτημα είναι η εξάρτηση όλων των εργασιών από τον server στον οποίο είναι ανεπτυγμένο: σε περίπτωση τεχνικής βλάβης ή αδυναμίας πρόσβασης η ομάδα δεν μπορεί να εργαστεί.

Git

Το Git ακολουθεί μια ριζοσπαστική προσέγγιση που διαφέρει πολύ από το CVS και το SVN. Οι αρχικές έννοιες για το Git ήταν να καταστήσουν ένα ταχύτερο, κατανεμημένο σύστημα ελέγχου αναθεώρησης το οποίο θα ανοιχτούσε ανοιχτά τις συμβάσεις και τις πρακτικές που χρησιμοποιούνται στο CVS. Είναι κυρίως σχεδιασμένο για Linux και καταγράφει τις υψηλότερες ταχύτητες του εκεί.
Καθώς δεν υπάρχει κεντρικός server, το Git δεν προσφέρεται σε μεμονωμένα έργα προγραμματιστών ή μικρές ομάδες, καθώς ο κώδικας μπορεί να μην είναι αναγκαστικά διαθέσιμος όταν χρησιμοποιείται από υπολογιστή υπολογιστή που δεν έχει πρόσβαση στο αποθετήριο. Υπάρχουν λύσεις για αυτό το πρόβλημα και κάποιοι βλέπουν τη βελτιωμένη ταχύτητα του Git ως αξιοπρεπή εμπόδιο για την ταλαιπωρία. Το Git έρχεται επίσης εξοπλισμένο με μια μεγάλη ποικιλία εργαλείων για να βοηθήσει τους χρήστες να πλοηγηθούν στο σύστημα ιστορικού των αρχείων. Κάθε επίβλεψης της πηγής ενός αρχείου περιέχει ολόκληρο το δέντρο ιστορικού.

2.7 Τεχνολογίες ανάπτυξης web εφαρμογών

2.7.1 Τεχνολογίες Ανάπτυξης Back – end (Business Logic)

.NET Framework

Το .Net Framework είναι μια πλατφόρμα ανάπτυξης λογισμικού που αναπτύχθηκε από τη Microsoft. Ως βασικό στόχο είχε τη δημιουργία εφαρμογών οι οποίες θα εκτελούνταν

στο λειτουργικό σύστημα Windows. Η πρώτη έκδοση .Net Framework κυκλοφόρησε το 2002 και ονομάστηκε .Net framework 1.0. Τρέχουσα έκδοση του είναι η 4.7.1. Το .Net framework μπορεί να χρησιμοποιηθεί για τη δημιουργία εφαρμογών που βασίζονται σε Desktop αλλά και Web λειτουργικότητα.[8] Βασικές γλώσσες που υποστηρίζει είναι η Visual Basic και η C #. Για την εφαρμογή "A Full CI Machine", η οποία είναι Desktop επιλέχθηκε το .NET Framework 4.6.1.

.NET Core

Το .NET Core είναι μια πλατφόρμα ανάπτυξης γενικού σκοπού που υποστηρίζεται από τη Microsoft και την κοινότητα .NET στο GitHub. Είναι πολλαπλής πλατφόρμας (υποστηρίζει Windows, macOS και Linux) και μπορεί να χρησιμοποιηθεί για την κατασκευή εφαρμογών για συσκευές, cloud και IoT. Είναι ενημερωμένο για ασφάλεια και ποιότητα αρκετές φορές το χρόνο, συνήθως μηνιαία. Το σπουδαιότερο χαρακτηριστικό του είναι ότι αποτελεί λογισμικό ανοικτού κώδικα. Για τις ανάγκες ανάπτυξης του API της εφαρμογής TravelStories χρησιμοποιήθηκε το .NET Core 2.2. [7]

Μελλοντική συγχώνευση των δύο τεχνολογιών

Όπως ανακοίνωσε η Microsoft το Μάιο του 2019, η επόμενη έκδοση του NET. Core θα ονομάζεται .NET 5 και θα υποστηρίζει όλες τις τεχνολογίες .NET Core και .NET Framework. Θα είναι cross-platform και εκτεινόμενη ημερομηνία πρώτης έκδοσης είναι ο Νοέμβριος του 2020. [9]

2.7.2 Τεχνολογίες Ανάπτυξης Front-end (UI)

Angular 6+

Η Angular, που αναπτύχθηκε από την Google, κυκλοφόρησε για πρώτη φορά το 2010, καθιστώντας την ως την παλαιότερη των σύγχρονων τεχνολογιών. Πρόκειται για ένα framework της JavaScript βασισμένο σε TypeScript. Η σημαντικότερη αλλαγή συνέβη το 2016 όταν και εκδόθηκε η Angular 2 (αφαιρώντας την κατάληξη JS από το αρχικό της όνομα (AngularJS)). Πλέον η Angular 2+ εκδίδεται σε ανανεωμένες και βελτιωμένες εκδόσεις περίπου ανά έξι μήνες και η τελευταία σταθερή έκδοση είναι η Angular 7, η οποίο κυκλοφόρησε τον Οκτώβριο του 2018.

Η Angular θεωρείται το πιο ώριμο από τα frameworks, έχει καλή υποστήριξη όσον αφορά τους συντελεστές της και είναι ένα πλήρες πακέτο.[12] Ωστόσο, απαιτεί προγραμματιστικό υπόβαθρο για όσους ξεκινούν την εκμάθηση της με αποτέλεσμα να δυσκολεύει και να αποθαρρύνει σχετικά άπειρους προγραμματιστές. Η Angular είναι μια καλή επιλογή για εταιρείες με μεγάλες ομάδες και προγραμματιστές που χρησιμοποιούν ήδη την TypeScript. Για τις ανάγκες ανάπτυξης της εφαρμογής TravelStories επιλέχθηκε η Angular 6, μιας και παρέχει σημαντική υποστήριξη στο διαδίκτυο και επίσης κάνει χρήση της Typescript.

React

Το React, το οποίο αναπτύχθηκε από το Facebook, κυκλοφόρησε αρχικά το 2013. Εκτός από το Facebook χρησιμοποιείται εκτεταμένα και σε άλλα μέσα κοινωνικής δικτύωσης (Instagram, WhatsApp κ.α.). Η τρέχουσα σταθερή έκδοση είναι η 16.X και κυκλοφόρησε το Νοέμβριο του 2018.

Το React έχει πλέον μεγαλώσει και υποστηρίζει τη χρήση πολλών προσθέτων που έχουν αναπτυχθεί για τη βελτίωση του. Σύμφωνα με μετρήσεις σταδιακά αποκτά μεγάλη αποδοχή. Το React μοιάζει με μια καλή επιλογή για κάποιον που θέλει να ξεκινήσει την ανάπτυξη web εφαρμογών με χρήση JavaScript, καθώς παρέχει μεγάλη ευελιξία.[12] Η δυνατότητα ενσωμάτωσης με άλλα πλαίσια παρέχει απρόσκοπτα ένα μεγάλο πλεονέκτημα για όσους επιθυμούν κάποια ευελιξία στον κώδικα τους.

Vue

Το Vue, επίσης γνωστό ως Vue.js, είναι το νεότερο από τα τρία frameworks. Αναπτύχθηκε από τον πρώην υπάλληλο της Google Evan You το 2014. Την τελευταία δυετία, το Vue παρουσίασε μεγάλη άνοδο, παρόλο που δεν έχει την υποστήριξη μιας μεγάλης εταιρείας. Η τρέχουσα σταθερή έκδοση είναι η 2.17, που κυκλοφόρησε τον Αύγουστο του 2018. Οι συνεισφέροντες της Vue υποστηρίζονται από την Patreon.[12] Το Vue 3, που βρίσκεται αυτή τη στιγμή στη φάση της ανάπτυξης, σχεδιάζει να μεταβεί στην TypeScript.

2.7.3 Τεχνολογίες Βάσεων Δεδομένων

SQLite

Η SQLite είναι ένα αυτόνομο RDBMS βασισμένο σε αρχεία, ανοικτού κώδικα, γνωστό για τη φορητότητα, την αξιοπιστία και την ισχυρή απόδοση του, ακόμα και σε περιβάλλοντα χαμηλής μνήμης. Οι συναλλαγές της είναι συμβατές με το πρωτόκολλο ACID (***A**tomicity, **C**onsistency, **I**solation, **D**urability*), ακόμη και σε περιπτώσεις που το σύστημα καταρρεύσει ή υποστεί διακοπή ρεύματος.

Με την SQLite, κάθε διαδικασία που έχει πρόσβαση στη βάση δεδομένων διαβάζει και γράφει απευθείας στα αρχεία της βάσης. Αυτό απλοποιεί σημαντικά τη διαδικασία εγκατάστασης της SQLite, καθώς εξαλείφει κάθε ανάγκη παραμετροποίησης. Ομοίως, δεν υπάρχουν απαραίτητες ρυθμίσεις για προγράμματα που θα χρησιμοποιούν τη βάση δεδομένων SQLite: το μόνο που χρειάζονται είναι η πρόσβαση στο δίσκο.

Η SQLite είναι δωρεάν λογισμικό ανοιχτού κώδικα και δεν απαιτείται ειδική άδεια χρήσης του. Ωστόσο, διατίθενται πολλές επί πληρωμή επεκτάσεις που βοηθούν στη συμπίεση και την κρυπτογράφηση. Επιπλέον προσφέρονται διάφορα αναβαθμισμένα πακέτα υποστήριξης χρηστών, το καθένα με ετήσιο τέλος.

PostgreSQL

Η PostgreSQL, γνωστή και ως Postgres, θεωρείται η "πιο προηγμένη βάση δεδομένων ανοιχτού κώδικα στον κόσμο". Δημιουργήθηκε με στόχο να είναι ιδιαίτερα επεκτάσιμη και να συμμορφώνεται με τα πρότυπα. Η PostgreSQL είναι μια βάση δεδομένων σχεσιακών αντικειμένων, που σημαίνει ότι αν και είναι πρωτίστως μια σχεσιακή βάση δεδομένων, περιλαμβάνει επίσης χαρακτηριστικά όπως η κληρονομιά πίνακα και η

υπερφόρτωση των λειτουργιών, τα οποία συσχετίζονται συχνότερα με βάσεις δεδομένων αντικειμένων.

Η Postgres είναι ικανή να χειρίζεται αποτελεσματικά πολλαπλά tasks ταυτόχρονα. Αυτό επιτυγχάνεται χωρίς κλειδαριές ανάγνωσης χάρη στην εφαρμογή του Συστήματος Ελέγχου Συνδρομικότητας Πολλαπλών Πολλαπλών Δεδομένων (MVCC), το οποίο εξασφαλίζει την ατομικότητα, τη συνέπεια, την απομόνωση και την ανθεκτικότητα των συναλλαγών του, γνωστή και ως συμμόρφωση με το ACID.

Η PostgreSQL δεν χρησιμοποιείται τόσο ευρέως όσο η MySQL, αλλά εξακολουθούν να υπάρχουν πολλά εργαλεία και βιβλιοθήκες τρίτων που έχουν σχεδιαστεί για να απλοποιήσουν τις διάφορες προγραμματιστικές εργασίες με τη PostgreSQL, συμπεριλαμβανομένων των pgAdmin και Postbird.

MySql

Σύμφωνα με την ταξινόμηση DB-Engines, η MySQL υπήρξε το δημοφιλέστερο RDBMS ανοιχτού κώδικα από τότε που ο ιστότοπος άρχισε να παρακολουθεί τη δημοτικότητα των διάφορων βάσεων δεδομένων (2012). Είναι ένα πλούσιο σε χαρακτηριστικά προϊόν που διευκολύνει πολλούς από τους μεγαλύτερους ιστοτόπους και εφαρμογές στον κόσμο όπως τα Netflix και Spotify. Ξεκινώντας με τη MySQL είναι σχετικά απλή, χάρη σε μεγάλο βαθμό στην τεκμηριωμένη τεκμηρίωσή της και στη μεγάλη κοινότητα προγραμματιστών, καθώς και στην αφθονία των πηγών που σχετίζονται με τη MySQL σε απευθείας σύνδεση.

Η MySQL σχεδιάστηκε για ταχύτητα και αξιοπιστία, σε βάρος της πλήρους τήρησης της τυπικής SQL. Η MySQL έχει εμπνεύσει πληθώρα εφαρμογών, εργαλείων και ολοκληρωμένων βιβλιοθηκών από τρίτους, οι οποίες επεκτείνουν τη λειτουργικότητά της και βοηθούν στην ευκολότερη συνεργασία. Ορισμένα από τα ευρύτερα χρησιμοποιούμενα από αυτά τα εργαλεία τρίτου μέρους είναι τα phpMyAdmin, DBeaver και HeidiSQL. Στη συγκεκριμένη εφαρμογή επιλέχθηκε η MySQL. Για τις ανάγκες της ανάπτυξης της εφαρμογής TravelStories χρησιμοποιήθηκε η MySQL, λόγω της ευκολίας που παρέχει στην αλληλεπίδραση με διάφορες τεχνολογίες.

3 Ανάπτυξη των δύο εφαρμογών

3.1 Διαδικτυακή Εφαρμογή

3.1.1 Περιγραφή – θεματολογία διαδικτυακής εφαρμογής

Η εφαρμογή διαδικτυακού χαρακτήρα με επαγγελματικό υπόβαθρο (business logic) ονομάζεται Travel Stories και έχει ως βασικό στόχο την δημιουργία ενός χρονολόγιου το οποίο αποτελείται από μία ή περισσότερες δομές που περιέχουν τα βασικά χαρακτηριστικά μιας εκδρομής, ταξιδιού ή εξόρμησης.

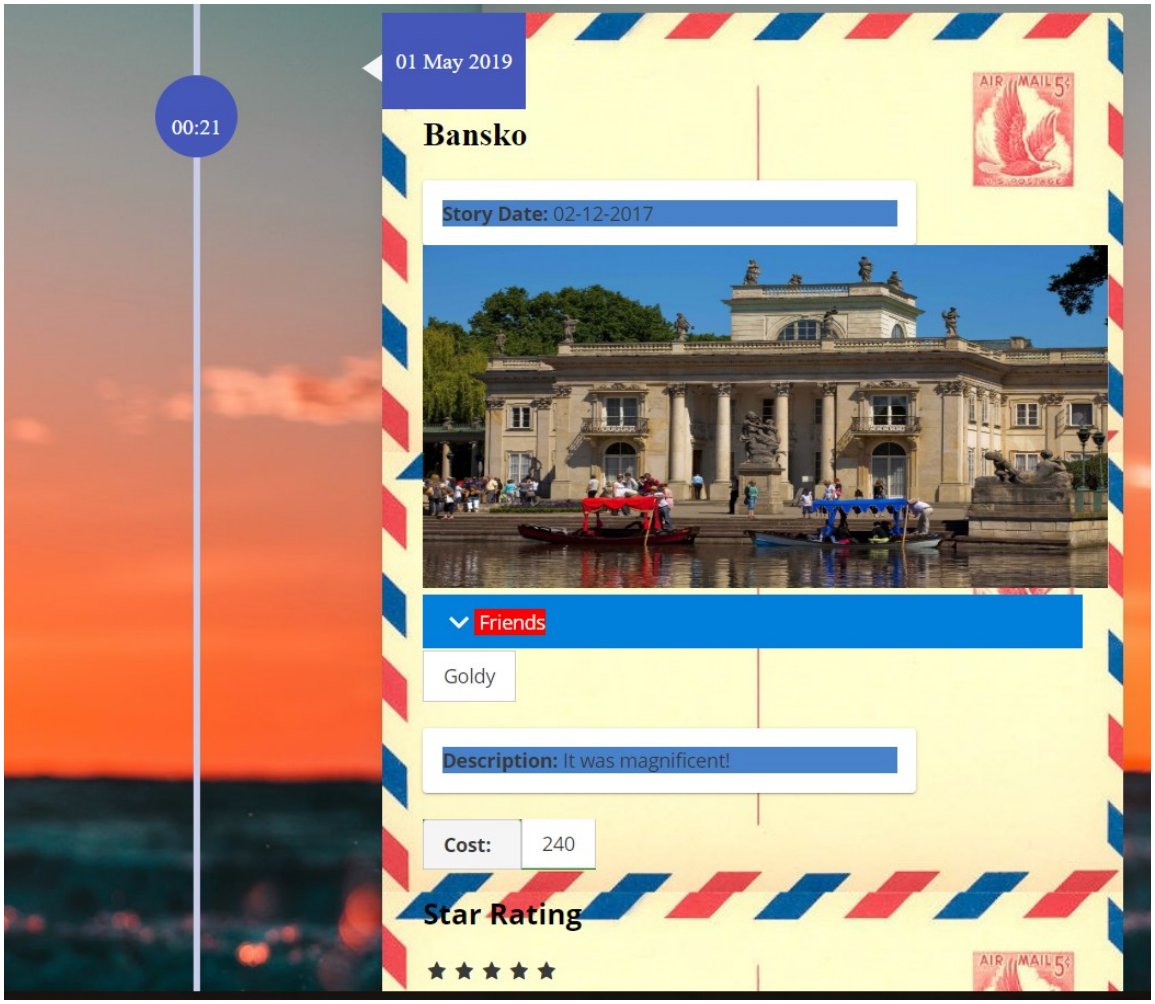
Σε αυτό το πρώιμο στάδιο της, ο χρήστης έχει τη δυνατότητα να εισέρχεται στην εφαρμογή με συνθηματικό (log in) να παρακολουθεί τις ιστορίες που έχει καταγράψει στο παρελθόν με χρονολογική σειρά και να προσθέτει νέες. Κάθε ιστορία περιέχει τοποθεσία, σύντομη περιγραφή, ημερομηνία, άτομα τα οποία συμμετείχαν, κόστος, βαθμολόγηση από το χρήστη και τέλος δυνατότητα μεταφόρτωσης μιας εικόνας.

Εικόνα 1: Οθόνη εισόδου στη διαδικτυακή εφαρμογή

3.1.2 Ανάπτυξη διαδικτυακής εφαρμογής

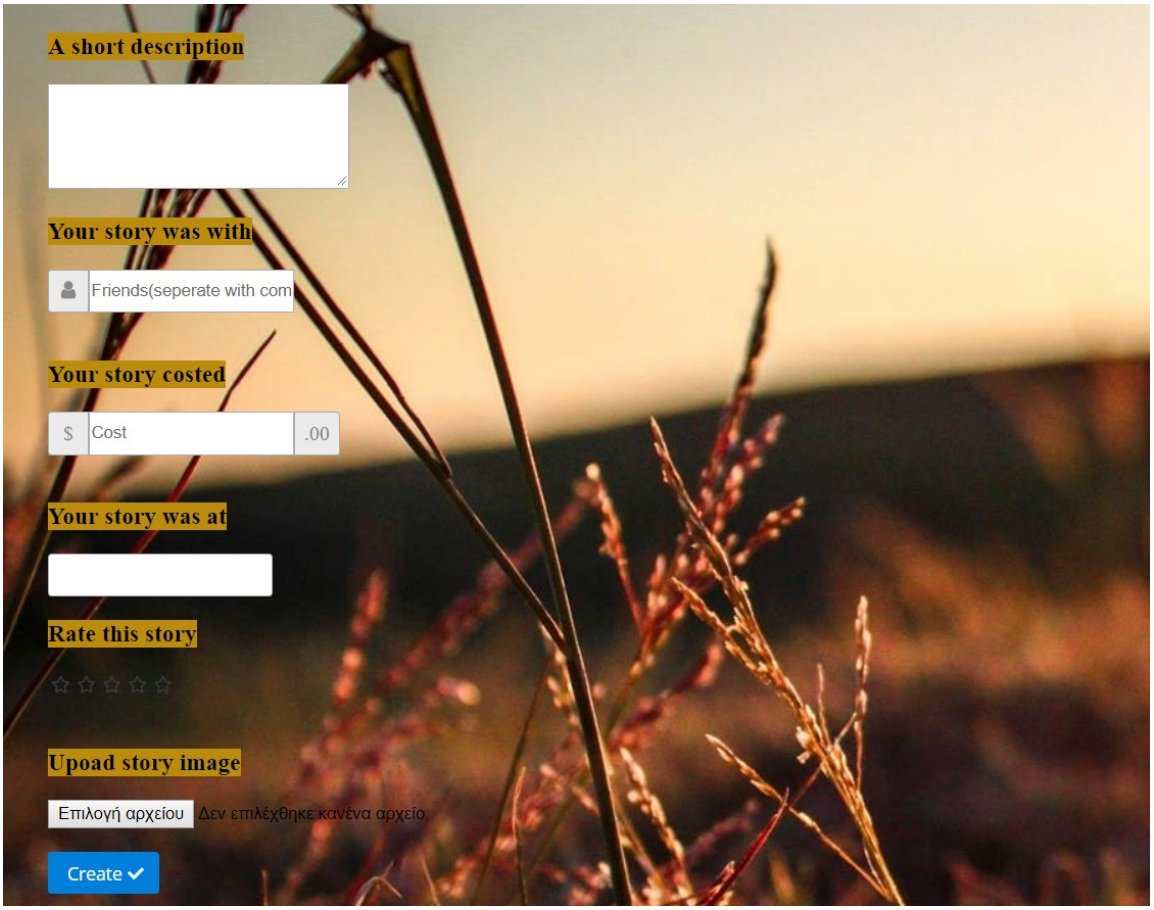
Η εφαρμογή αποτελείται από ένα .NET Core API στο back-end επίπεδο της, αναπτυγμένο στη γλώσσα προγραμματισμού C#. Εκεί υπάρχουν μέθοδοι οι οποίες “ακούν” rest κλήσεις από το UI, αλληλεπιδρούν με τη βάση δεδομένων και επιστρέφων πληροφορία στο front-end με σκοπό να επεξεργαστεί και να εμφανιστεί όμορφα στον τελικό χρήστη. Εκεί συναντούμε επίσης τα μοντέλα, δηλαδή κλάσεις με ιδιότητες που αντιστοιχούν σε πίνακες στη βάση αλλά και σε αντίστοιχα μοντέλα στο front-end. Για

παράδειγμα υπάρχει το μοντέλο User με ιδιότητες το ID, Username, Password, YearOfBirth κλπ. Αξίζει να σημειωθεί ότι το περιβάλλον ανάπτυξης του κώδικα ήταν το Visual Studio 2017.



Εικόνα 2: Μια εγγραφή του χρονολογίου που περιγράφει το αντίστοιχο ταξίδι

Στη βάση δεδομένων συναντάμε πίνακες οι οποίοι έχουν αντιστοιχία με τα μοντέλα της εφαρμογής και χρησιμοποιούνται για την αποθήκευση δεδομένων που εισάγει ο χρήστης. Η τεχνολογία της βάσης είναι MySQL και η επικοινωνία με το back-end γίνεται κατά βάση με SQL scripts ενσωματωμένα σε κώδικα της C#. Για την ανάπτυξη της βάσης χρησιμοποιήθηκε η πλατφόρμα MySQL Workbench 8.0



Εικόνα 3: Οθόνη δημιουργίας νέας ιστορίας

Στο front-end κομμάτι, χρησιμοποιήθηκε Angular 6+ (Typescript+Html+CSS), καθώς και η γραμμή εντολών της Angular, το Angular CLI για την αυτοματοποιημένη ανάπτυξη μεγάλων δομών (modules, components).[16] Εκεί διαβάζονται τα δεδομένα από το UI και στέλνονται μέσω REST κλήσεων στο back-end με απώτερο σκοπό να αποθηκευτούν στη βάση για μελλοντική χρήση. Επίσης γίνονται αντίστροφες κλήσεις στο back-end και από εκεί στη βάση ώστε να ανακτηθούν δεδομένα, να επεξεργαστούν και τελικά να παρουσιαστούν στο χρήστη. Το front-end κομμάτι αναπτύχθηκε στην πλατφόρμα Visual Studio Code.

3.2 Εφαρμογή Συνεχούς Ενσωμάτωσης

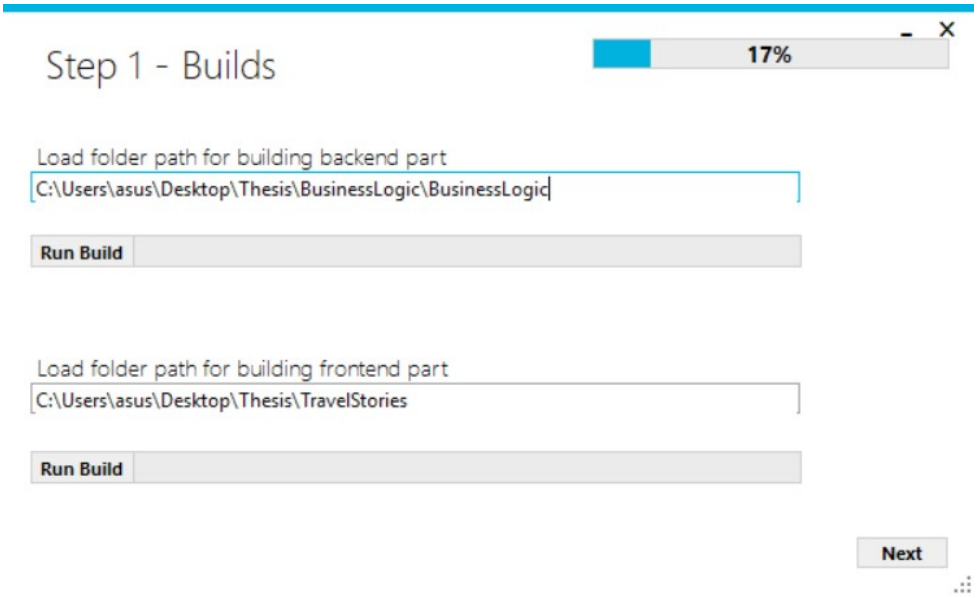
3.2.1 Περιγραφή Εφαρμογής Συνεχούς Ενσωμάτωσης

Η εφαρμογή που εκτελεί εντολές που απαρτίζουν έναν κύκλο συνεχούς ενσωμάτωσης ονομάζεται “A Full CI Machine” και κατά κύριο λόγο ενεργεί σε τοπικό επίπεδο πριν ετοιμάσει τα αρχεία και τα μεταφορτώσει στον server. Ο χρήστης περνάει όλα τα βήματα ενός κύκλου συνεχούς ενσωμάτωσης μέσα από έξι οθόνες, καθώς επίσης και της παραμετροποίησης από μία ακόμη.

3.2.2 Ανάπτυξη Εφαρμογής Συνεχούς Ενσωμάτωσης

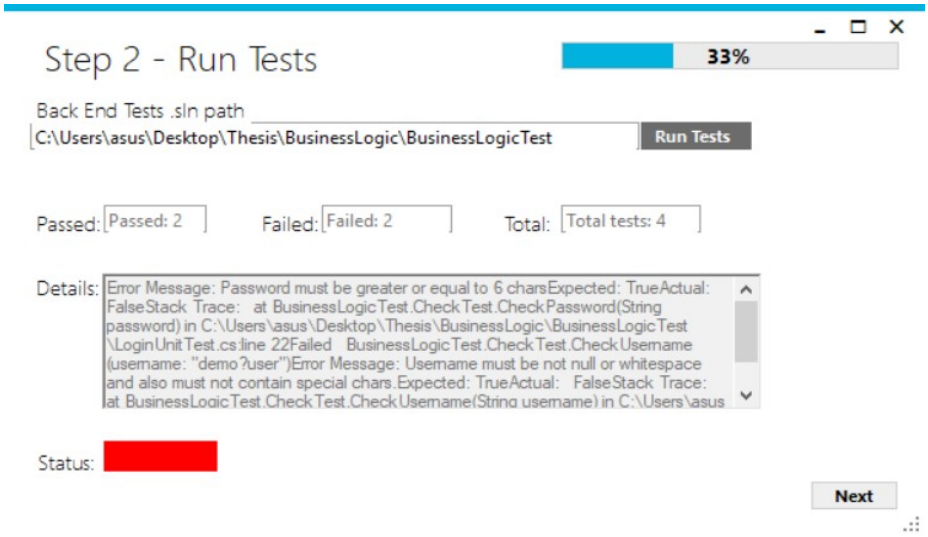
Η εφαρμογή “A Full CI Machine” είναι ανεπτυγμένη εξ ολοκλήρου σε C# (NET. Framework 4.6.1) όσον αφορά το back-end τμήμα της και σε C# με χρήση της τεχνολογίας .NET Winforms για το UI. Και τα δύο μέρη αναπτύχθηκαν μέσω του Visual Studio 2015.

Ο χρήστης στο πρώτο βήμα εκτελεί δύο εντολές build με χρήση της γραμμής εντολών (cmd), τη .dotnet build για το back-end και την ng server για το front-end, και οι δύο στα μονοπάτια αρχείων που του υποδεικνύει ο χρήστης. [6]



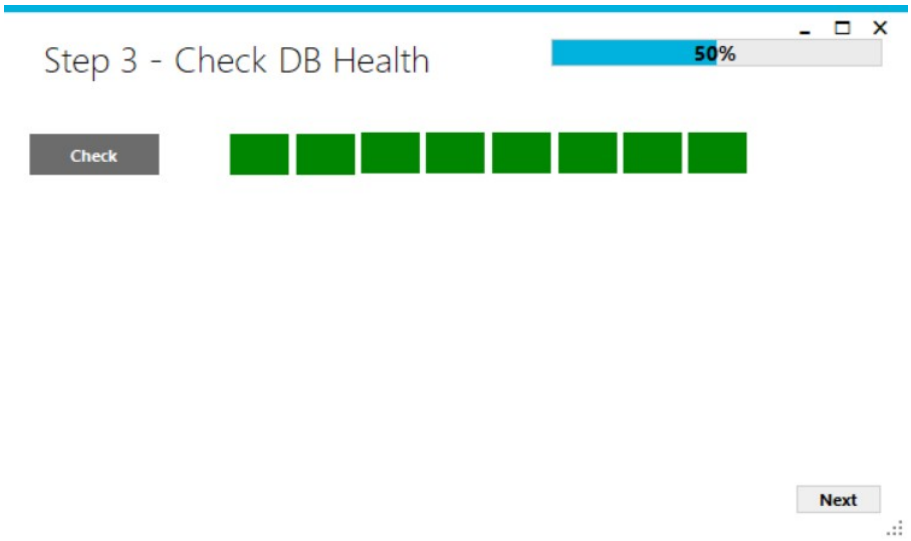
Εικόνα 4: Βήμα 1ο, εκτέλεση των builds

Στο δεύτερο βήμα ο χρήστης εκτελεί την εντολή dotnet test [6], στο σχετικό μονοπάτι αρχείων, με σκοπό να τρέξουν όλα τα back-end tests και να λάβει το αποτέλεσμα τους.



Εικόνα 5: Δεύτερο βήμα, εκτέλεση των back-end ελέγχων

Στο τρίτο βήμα επιχειρείται μια δοκιμαστική σύνδεση με τη βάση, με την εκτέλεση ενός δοκιμαστικού sql script μέσω της C#. Σε περίπτωση επιτυχίας πρασινίζουν τα αντίστοιχα κουτάκια, σε κάθε άλλη περίπτωση κοκκινίζουν.

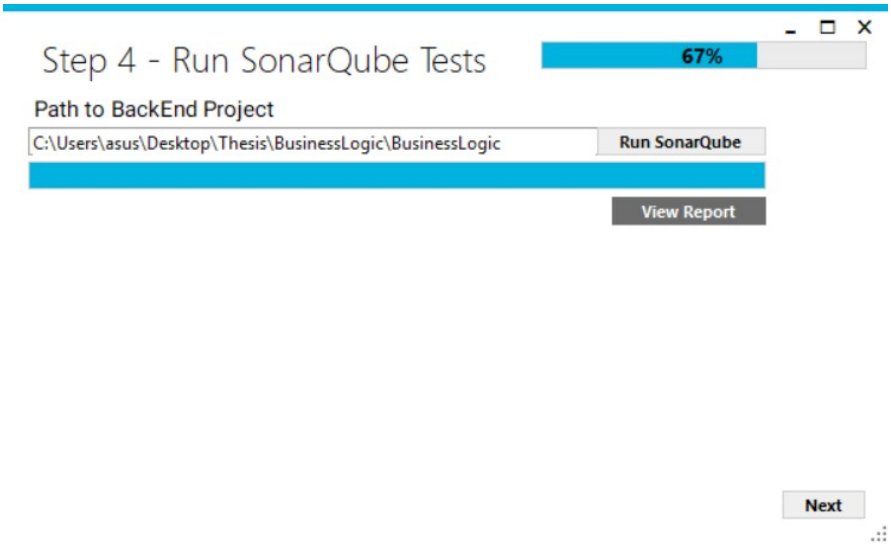


Εικόνα 5: Βήμα 3ο, έλεγχος συνδεσιμότητας με βάση δεδομένων

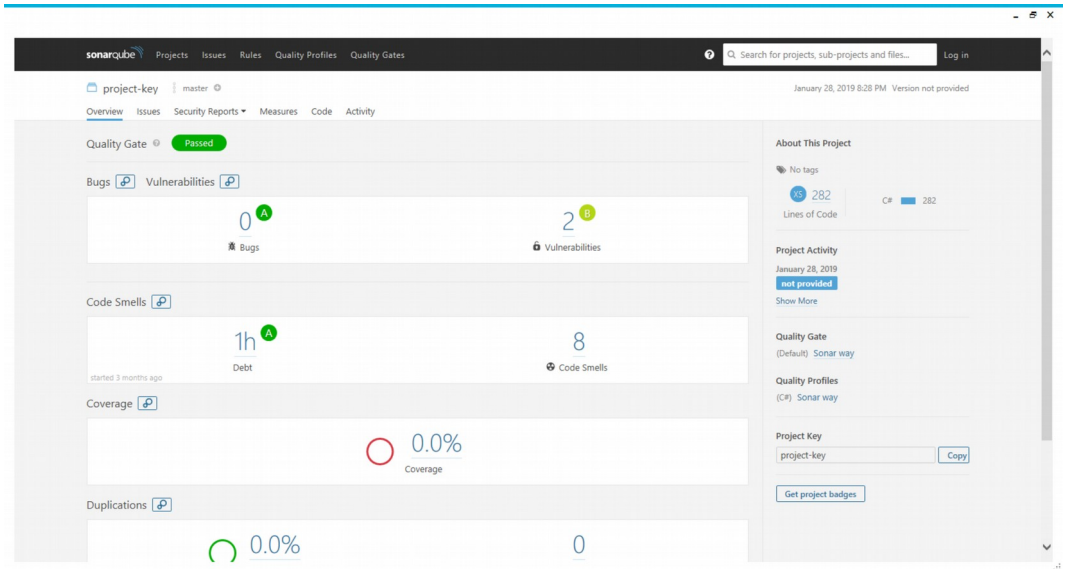
Στο επόμενο βήμα, έχουμε την αξιολόγηση του κώδικα με τη βοήθεια του εργαλείου στατικής ανάλυσης κώδικα SonarQube. Ο χρήστης δηλώνει σε ποιο project θέλει να

τρέξει του ελέγχους και όταν πατήσει το κουμπί “Run SonarQube” εκτελείται μέσω γραμμής εντολών (cmd) η εντολή dotnet sonarscanner end. Για να επιτευχθεί ωστόσο η ανάλυση, απαιτείται να τρέχει το SonarQube ως service στο συγκεκριμένο υπολογιστή. Όταν το report είναι έτοιμο με το πάτημα του κουμπιού “View Report” η εφαρμογή

ανοίγει έναν δικό της φυλλομετρητή (browser) και προβάλει τα αποτελέσματα.

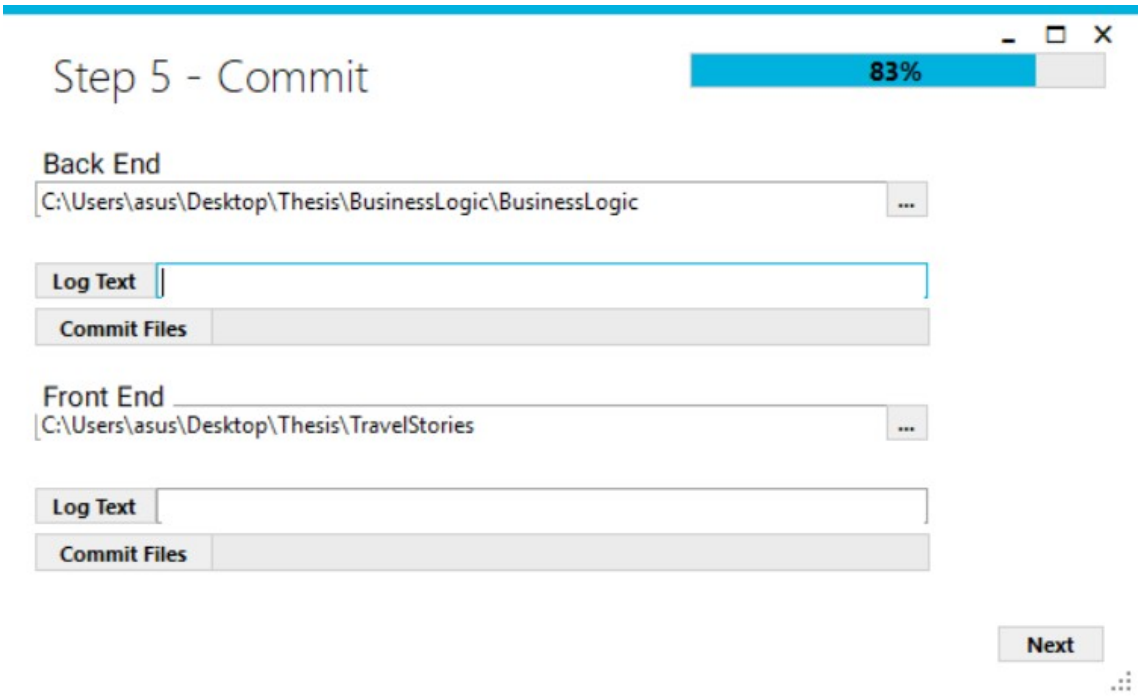


Εικόνα 6: Βήμα 4ο, δυνατότητα ποιοτικής ανάλυσης του κώδικα



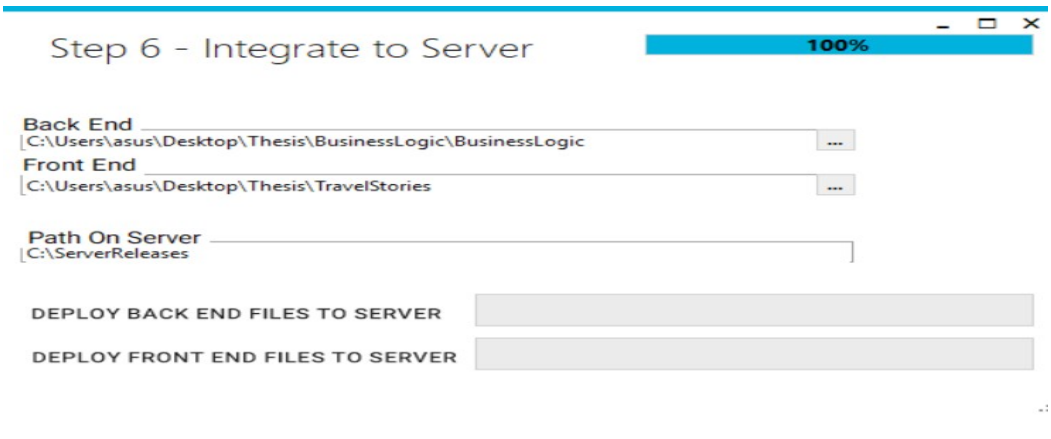
Εικόνα 7:Βήμα 4ο, προεπισκόπηση της αναφοράς ποιότητας του κώδικα

Στο προτελευταίο βήμα ο χρήστης κάνει ενσωμάτωση/μεταφόρτωση των αλλαγών ή προσθηκών του στον svn server. Ουσιαστικά κρατείται ιστορικό αλλαγών και εκδόσεων για κάθε αρχείο και ο χρήστης μέσω της εντολής “Commit” ανεβάζει στον svn server τις τελευταίες αλλαγές του. Αυτό βοηθάει σημαντικά στη διατήρηση ιστορικού των εκδόσεων του κώδικα, στον εντοπισμό λαθών, στην αλλαγή εκδόσεων αλλά κυρίως στον παράλληλο προγραμματισμό μεταξύ δύο ή περισσότερων προγραμματιστών. Για να δουλέψει ένας προγραμματιστής σε ένα αρχείο που κάνει αλλαγές ένας δεύτερο προγραμματιστής θα πρέπει να συγχρονιστούν οι εκδόσεις του συγκεκριμένου αρχείου στους υπολογιστές του μέσω των εντολών “Commit” και “Update”.



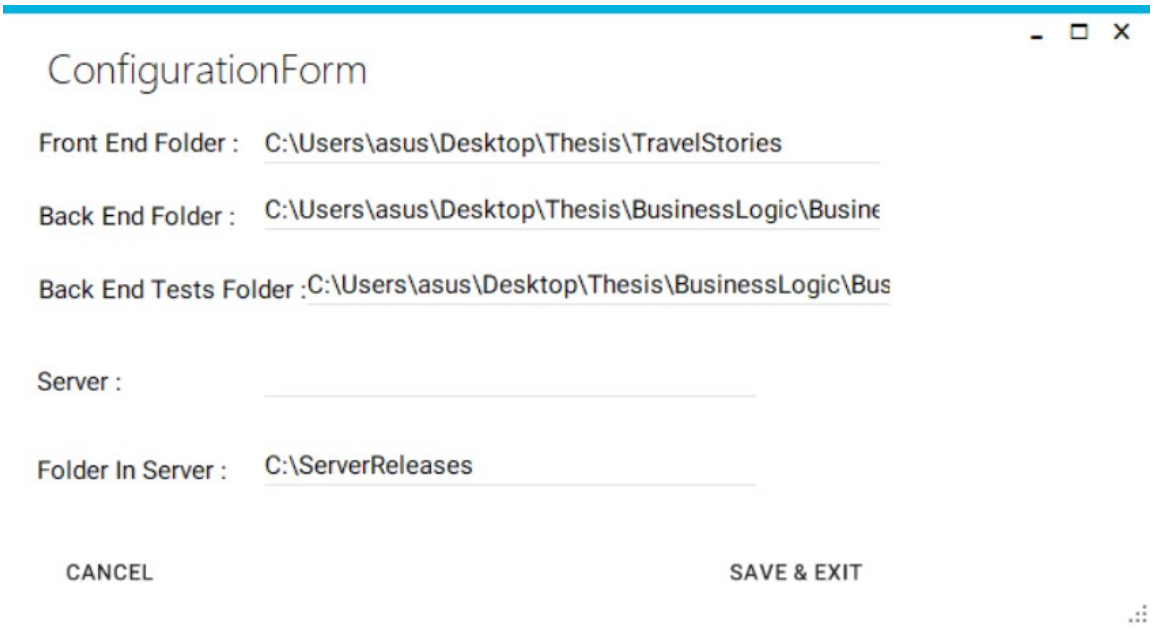
Εικόνα 8: Βήμα 5ο, Commit των αλλαγών ή προσθηκών στον SVN Server

Στο τελευταίο βήμα ο χρήστης ορίζει τους φακέλους που περιέχουν τα back-end και front-end αρχεία, καθώς και τον κοινόχρηστο φάκελο δικτύου στον οποίο έχει πρόσβαση ο “Release Server”, για να αντιγραφούν εκεί και να έχει πρόσβαση σε αυτά.



3.3 Επικοινωνία των δύο Εφαρμογών

Η επικοινωνία των δύο εφαρμογών επιτυγχάνεται μέσω της οθόνης Configuration στην εφαρμογή A Full CI Machine. Εκεί παραμετρικά ο χρήστης ορίζει τις διαδρομές των αρχείων, των φακέλων του έργου, στην προκειμένη της εφαρμογής Travel Stories καθώς και τον κοινό φάκελο δικτύου στον οποίο έχουν πρόσβαση οι τοπικοί υπολογιστές και ο server με σκοπό σε επόμενο βήμα το Jenkins να έχει λάβει έτοιμα τα αρχεία με σκοπό να "τρέξει" κάποια build στον server. Με την αποθήκευση των απαραίτητων στοιχείων κάθε βήμα του CI έχει ήδη έτοιμα τα στοιχεία που χρειάζεται για να προβεί σε ενέργειες, π.χ. build, commit κλπ, χωρίς να χρειαστεί ο χρήστης να τα δηλώνει σε κάθε βήμα. Ωστόσο ο χρήστης της εφαρμογής A Full CI Machine έχει τη δυνατότητα σε κάθε βήμα να υπερκαλύπτει τη παραμετροποίηση αυτή με νέα δικά του στοιχεία, αν βεβαίως κριθεί αναγκαίο.



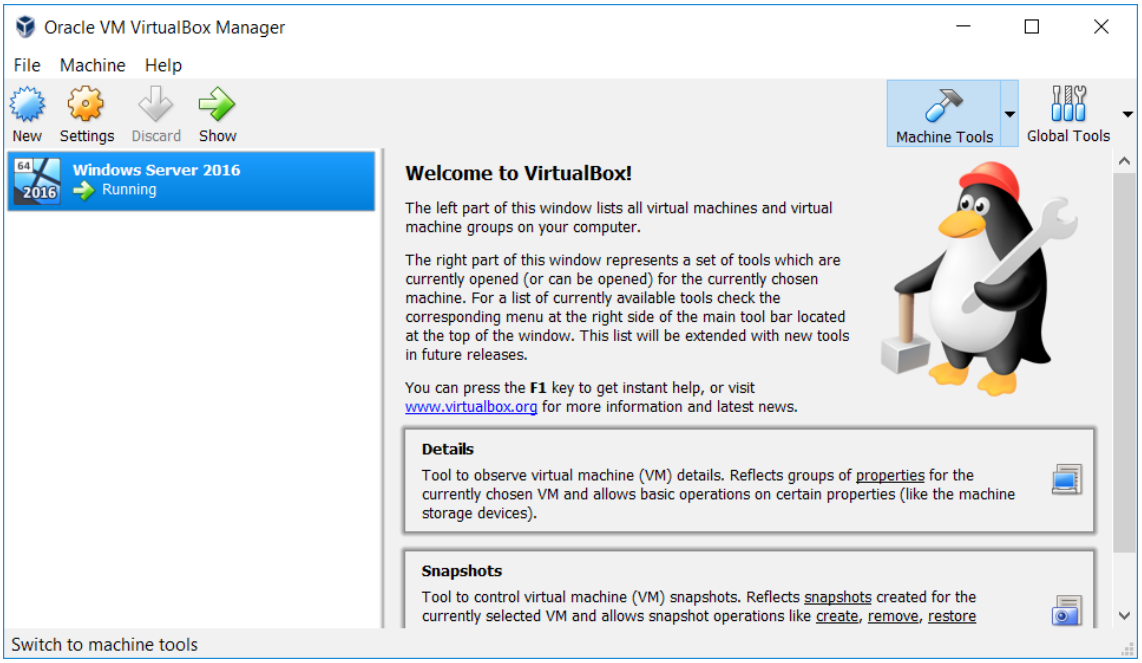
Εικόνα 10: Φόρμα παραμετροποίησης

3.4 Ανάπτυξη υπάρχοντος συστήματος CI στον server.

3.4.1 Δημιουργία Server σε Virtual Box.

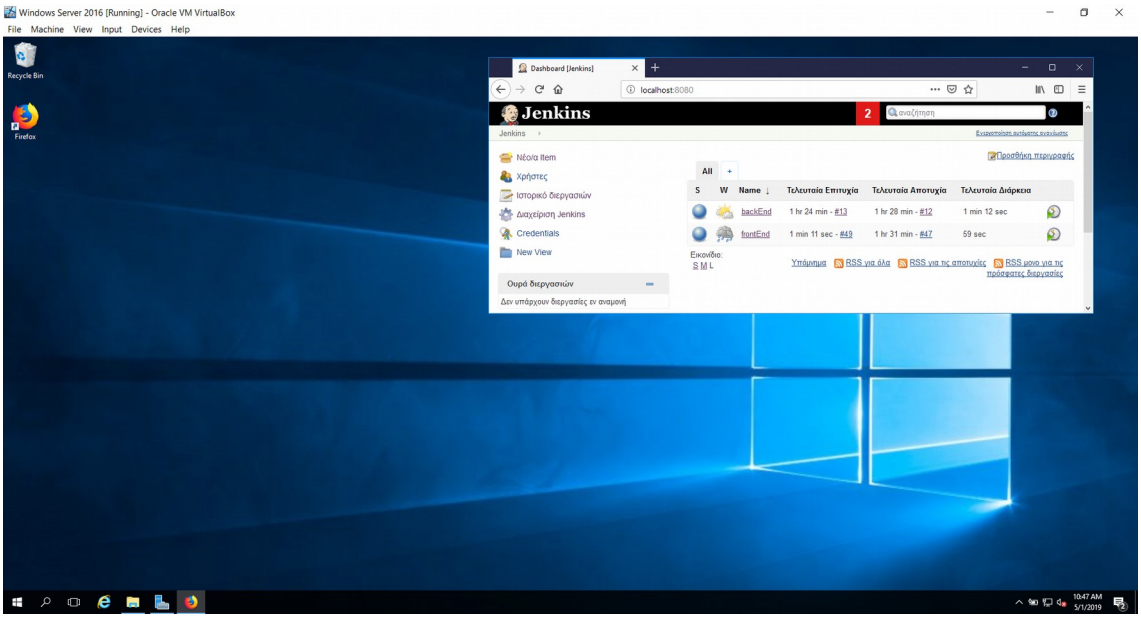
Για τις ανάγκες της συνεχούς ενσωμάτωσης ενός έργου λογισμικού είναι αναγκαία ή ύπαρξη τουλάχιστον ενός server όπου θα ενσωματώνονται όλες οι τελευταίες αλλαγές

των προγραμματιστών στον κώδικα. Σε αυτόν τον server έχουν δυνατότητα σύνδεσης και επικοινωνίας όλοι οι υπολογιστές του τοπικού δικτύου. Συνήθως ο υπολογιστής αυτός ονομάζεται “Release Server” ή “Production Server” καθώς συγκεντρώνει το έργο λογισμικού στην τελική του μορφή όπως θα το χρησιμοποιεί ένας χρήστης. Επίσης για λόγους ασφαλείας ο χρήστης μιας διαδικτυακής εφαρμογής έχει πρόσβαση μόνο στον server και όχι στους υπολογιστές των προγραμματιστών που αναπτύσσουν την εφαρμογή.



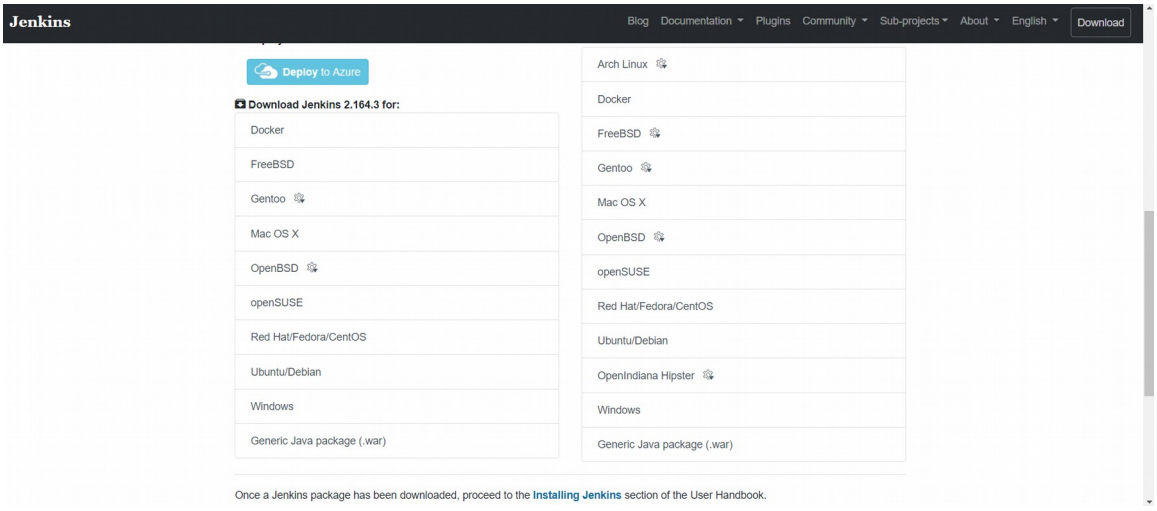
Εικόνα 11: Γραφικό περιβάλλον Oracle VM VitualBox

Για το λόγο αυτό, δημιουργήθηκε μια εικονική μηχανή με τη βοήθεια του Virtual Box της Oracle η οποία έχει ως λειτουργικό σύστημα το Windows Server 2016, όπου γίνονται deploy τα αρχεία μετά από κάθε κύκλο συνεχούς ενσωμάτωσης. Αυτό επιτυγχάνεται καθώς ο server ανήκει στο ίδιο δίκτυο με τον υπολογιστή του προγραμματιστή. Το Virtual Box παρέχει τη δυνατότητα δημιουργίας shared folders, δηλαδή φακέλων οι οποίοι θα ανήκουν στο δίκτυο και στους οποίους θα έχουν πρόσβαση όλοι οι υπολογιστές που ανήκουν σε αυτό.[10] Σε αυτούς τους φακέλους καταλήγουν τα αρχεία κώδικα, και παράλληλα έχει δικαιώματα ανάγνωσης, εγγραφής και εκτέλεσης το Jenkins. Έτσι τα τελικά build στον “Release Server” γίνονται με τη βοήθεια του Jenkins.



Εικόνα 12: VM, με λειτουργικό σύστημα το Windows 2016 Server και εγκατεστημένο το Jenkins

3.4.2 Εγκατάσταση και Παραμετροποίηση του Jenkins



Εικόνα 13: Ιστότοπος Jenkins CI

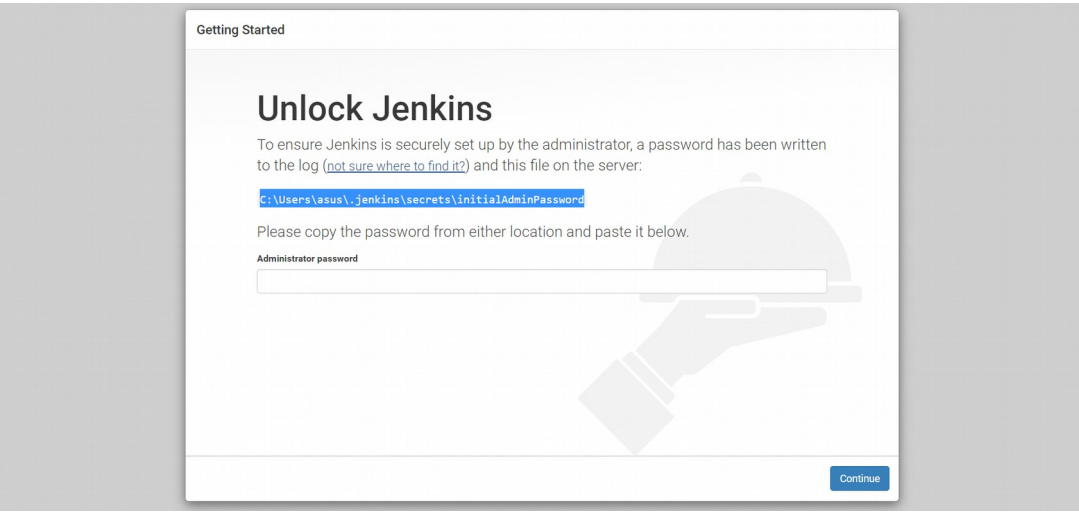
Όπως προαναφέρθηκε επιλέχτηκε η χρησιμοποίηση του Jenkins, για δύο βασικά του πλεονεκτήματα: τη δωρεάν χρήση του και τη δυνατότητα υποστήριξης πολλαπλών τεχνολογιών. Αρχικό βήμα είναι η μετάβαση στην ιστοσελίδα του, και η λήψη της αντίστοιχης έκδοσης με το λειτουργικό σύστημα στο οποίο θα λειτουργήσει. Στην προκειμένη περίπτωση κατέβηκε η έκδοση για τα Windows. Έπειτα αποσυμπιέστηκε στο

αρχείο που λήφθηκε και εκτελέστηκε. Στα αρχικά βήματα επιλέχθηκε η επιλογή “Next” και έπειτα “Install”.

Εφόσον τελειώσει η εγκατάσταση το προεπιλεγμένο port στον υπολογιστή για να είναι προσβάσιμο το Jenkins είναι το 8080. Ωστόσο επειδή αυτό χρησιμοποιείται και από εφαρμογές που προϋπήρχαν στον server δίνεται η δυνατότητα να τεθεί ένα νέο port για το Jenkins, στη συγκεκριμένη περίπτωση το 9090.

Αυτό επιτυγχάνεται αν μέσω της γραμμής εντολών (cmd) εκτελεστούν οι παρακάτω δύο εντολές. **cd C:\Program Files (x86)\Jenkins**, η οποία μεταβαίνει στο μονοπάτι που εγκαταστάθηκε το Jenkins και έπειτα, **java -jar jenkins.war –httpPort=9090**.

Έπειτα από την εκτέλεση αυτών των εντολών, με απλή μετάβαση μέσω browser στο **localhost:9090**, και συμπλήρωση του κλειδιού από το αρχείο που υποδεικνύεται επιλέχθηκε η δυνατότητα “Install Suggested Plugins”.[3]



Εικόνα 14: Unlock Jenkins με κλειδί που υποδεικνύεται

Όταν τελείωσε η εγκατάσταση πατήθηκε το κουμπί “Continue As Admin”. [3]

Εικόνα 15: Επίλογή εγκατάστασης προσθέτων στο Jenkins CI

Μετά από την εγκατάσταση σειρά λαμβάνει η δημιουργία των δύο jobs μια για κάθε μέρος της εφαρμογής, με ονόματα front-end και back-end. Επιλέχθηκε από το μενού αριστερά το "Νέο Item". Στη συνέχεια συμπληρώθηκε το όνομα του και η πατήθηκε η επιλογή "Free Style Project".

Για κάθε ένα από τα δύο items συμπληρώθηκε στο "Build Triggers" → "Build Periodically" η δυνατότητα εκτέλεσης των build με χρονοπρογραμματισμό. Συμπληρώθηκαν οι τιμές **30 08 * * 1-5** και **00 09 * * 1-5** , ώστε να γίνονται ημερήσια builds από Δευτέρα ως και Παρασκευή 8 και 30 το πρωί και 9 το πρωί αντίστοιχα. Με αυτή τη δυνατότητα αν και εφόσον για παράδειγμα η ομάδα των προγραμματιστών ξεκινά να εργάζεται μετά τις 9 και 30 κάθε μέρα, έχουν προηγηθεί τα builds που περιλαμβάνουν τα deploy τις προηγούμενης μέρας και είναι ορατό το αποτέλεσμα τους. Αν κάτι δεν πάει καλά, οι προγραμματιστές ενημερώνονται και έχουν τη δυνατότητα να το φτιάξουν χωρίς να παραμένει εκτεθιμένο το σύστημα για πολλή ώρα.

Οι εντολές για κάθε build είναι ίδιας λογικής, δηλαδή build του κώδικα, αλλά διαφορετικής σύνταξης μιας και χρησιμοποιείται διαφορετική τεχνολογία (.NET Core και Angular 6) αντίστοιχα. Για το back-end item χρησιμοποιήθηκε η εντολή **pushd** \\ VBOXSVR\ServerReleases**FrontEnd** και **npm run ng build** και για το back-end

pushd \\VBOXSVR\ServerReleases\BackEnd\BusinessLogic και **dotnet build**. Ακολουθεί στο επόμενο κεφάλαιο εκτενής επεξήγηση τους.

4 Αλληλεπίδραση των δύο Εφαρμογών – Συνεχής Ενσωμάτωση

4.1 Ένας κύκλος συνεχούς ενσωμάτωσης

Προετοιμασία της Παραμετροποίησης

Το πρώτο βήμα που οφείλει να κάνει ένας χρήστης της εφαρμογής A Full CI Machine είναι να ελέγξει και αν χρειάζεται να τροποποιήσει την παραμετροποίηση, ώστε να προχωρήσουν αυτοματοποιημένα τα βήματα της συνεχούς ενσωμάτωσης. Τα στοιχεία που πρέπει να δώσουμε για τις ανάγκες της παραμετροποίησης είναι το μονοπάτι του φακέλου που υπάρχουν τα αρχεία του front-end και του back-end αντίστοιχα, το μονοπάτι του φακέλου που υπάρχουν τα αρχεία για τα tests στο back-end και τέλος το μονοπάτι στον κοινόχρηστο φάκελο δικτύου όπου θα γίνεται μεταφόρτωση των αρχείων στο τέλος κάθε κύκλου συνεχούς ενσωμάτωσης.

Βήμα 1. Build των δύο συστατικών μερών της εφαρμογής

Ουσιαστικά πρόκειται για ένα από τα πιο θεμελιώδη τμήματα ενός κύκλου συνεχούς ενσωμάτωσης. Ο προγραμματιστής λαμβάνει έτοιμα τα μονοπάτια των αρχείων των δύο μερών της εφαρμογής (back-end και front-end) και με το πάτημα του κουμπιού build εκτελεί την αντίστοιχη εντολή που περιλαμβάνει μεταγλώττιση και σύνδεση του κώδικα. Αυτός ο έλεγχος κυρίως εντοπίζει συντακτικά λάθη και σχεδόν ποτέ λογικά λάθη. Αν οι μπάρες προόδου (progress bar) λάβουν γαλάζιο χρώμα τα builds έχουν στεφθεί με επιτυχία. Σε διαφορετική περίπτωση ο χρήστης καλείται να ανοίξει το αντίστοιχο περιβάλλον ανάπτυξης κώδικα και να διαβάσει τα μηνύματα σφαλμάτων ώστε να εντοπίσει και να διορθώσει τα λάθη. Οι παραπάνω εντολές εκτελούνται από την εφαρμογή με τη βοήθεια της γραμμής εντολών.

Βήμα 2. Εκτέλεση των tests στο back-end

Σε αυτό το βήμα ο χρήστης καλείται μέσω της εφαρμογής να εκτελέσει όλα τα διαθέσιμα tests για το back-end μέρος της διαδικτυακής εφαρμογής. Αφού πατήσει το κουμπί “Run Tests” λαμβάνει μια σύντομη ενημέρωση η οποία περιλαμβάνει αριθμό συνολικών ελέγχων που εκτελέστηκαν, αριθμό επιτυχημένων και αριθμό αποτυχημένων. Σε περίπτωση που υπάρχει τουλάχιστον ένα αποτυχημένο test ο χρήστης λαμβάνει στην οθόνη του (“Details”) μια αρκετά αναλυτική περιγραφή του λάθους ώστε να βοηθηθεί και να διορθώσει σε μικρό χρόνο το αντίστοιχο σφάλμα. Επίσης δίπλα στο πεδίο

“Status” εμφανίζεται ένα χρώμα το οποίο αντιπροσωπεύει την κατάσταση στην οποία βρίσκεται ο κώδικα. Πράσινο αν όλοι οι έλεγχοι έχουν περάσει, κόκκινο αν απέτυχε έστω και ένας.

Βήμα 3. Έλεγχος κατάστασης βάσης δεδομένων

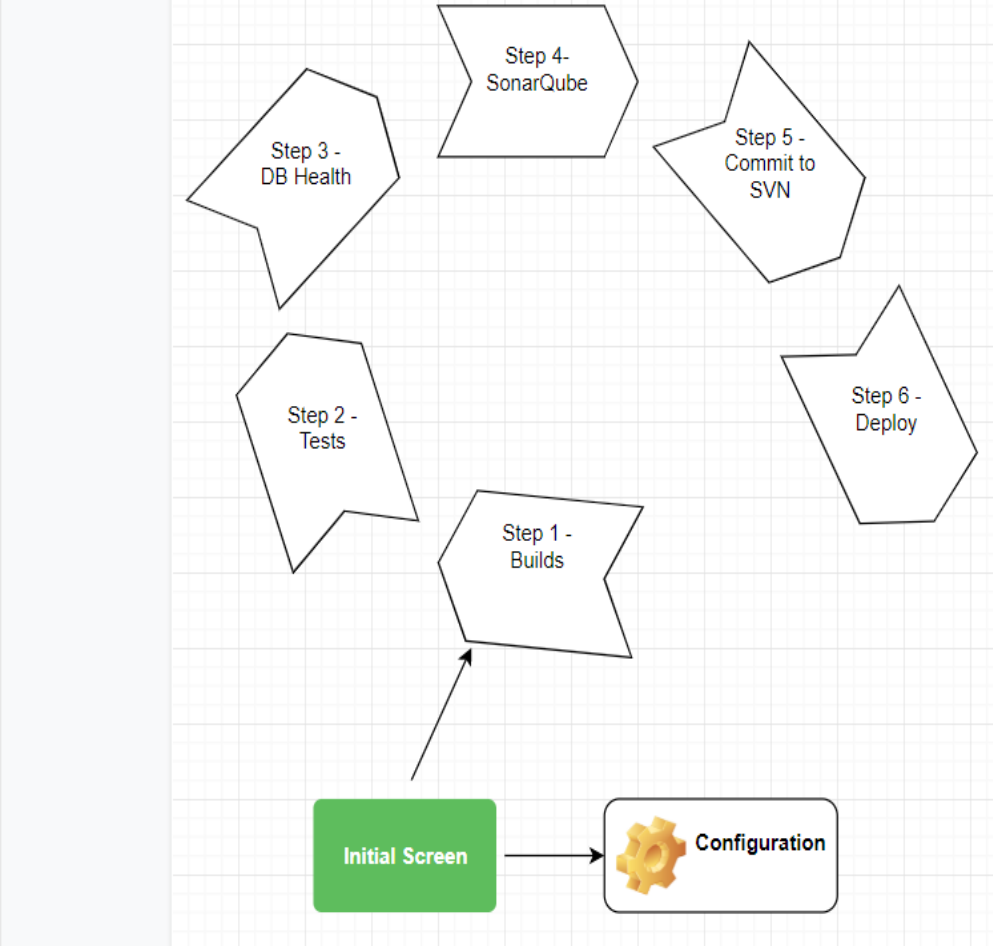
Αποτελεί ένα πολύ σύντομο αλλά σημαντικό βήμα καθώς με τη βοήθεια κάποιων απλών SQL εντολών η εφαρμογή ελέγχει αν η βάση δεδομένων είναι διαθέσιμη. Ο χρήστης δεν έχει πρόσβαση ή ενημέρωση σε αυτές τις εντολές και απλώς λαμβάνει το αποτέλεσμα στην οθόνη του. Σε περίπτωση που το χρώμα στα αντίστοιχα κουτάκια είναι κόκκινο θα πρέπει να ελέγξει τη βάση του, αν είναι πράσινο η βάση είναι κανονικά διαθέσιμη.

Βήμα 4. Στατική Ανάλυση του Κώδικα με τη χρήση SonarQube

Στο 4ο βήμα η εφαρμογή A Full CI Machine ελέγχει την ποιότητα του κώδικα και δημιουργεί μια αναφορά στην οποία παρουσιάζονται ενδιαφέρουσες μετρικές σχετικά με τον κώδικα με τη χρήση του εργαλείου SonarQube. Αναλυτικότερα, πατώντας το κουμπί “Run SonarQube” με τη βοήθεια γραμμής εντολών, η εφαρμογή δίνει στο εργαλείο SonarQube το μονοπάτι στο οποίο υπάρχουν τα αρχεία της εφαρμογής Travel Stories και αυτό εκτελώντας διάφορους ελέγχους και αναλύσει δημιουργεί μια αναφορά. Μόλις η αναφορά είναι έτοιμη ενεργοποιείται το κουμπί “View Report”. Πατώντας το, η εφαρμογή ανοίγει ένα νέο παράθυρο με δυνατότητες φυλλομετρητή (browser) στο οποίο υπάρχει η αναφορά με ένα αρκετά σημαντικό πλήθος λεπτομερειών, όπως γλώσσα συγγραφής του κώδικα, αριθμός σφαλμάτων, εκτιμώμενος χρόνος συνολικής βελτίωσης του κώδικα, συνολικός αριθμός γραμμών κώδικα κλπ. Αξίζει να σημειωθεί ότι με τη βοήθεια του SonarQube ο κώδικας μπορεί να γίνει πολύ πιο ποιοτικός, να αυξηθεί η ασφάλεια της εφαρμογής, καθώς και η δυνατότητα συντήρησης ή επέκτασης της. Οι έλεγχοι τρέχουν μόνο για το back-end τμήμα της εφαρμογής.

Βήμα 5. Commit αλλαγών ή προσθηκών στον SVN server

Σε



Εικόνα 16: Απεικόνιση ενός κύκλου συνεχούς ενσωμάτωσης, μέσω της εφαρμογής A Full CI Machine

αυτό το σημείο ο χρήστης θα πρέπει να αποθηκεύσει τις νέες αλλαγές και των δύο τμημάτων της εφαρμογής στον SVN server. Εκεί κρατείται ιστορικό για κάθε αρχείο και δυνατότητα επαναφοράς μεμονωμένου αρχείου ή του project συνολικά. Μέσω του κουμπιού Commit η εφαρμογή δημιουργεί έναν SVN Client και καλεί την αντίστοιχη μέθοδο Commit ώστε να μεταφορτωθεί η αλλαγή στον server. Αυτό προϋποθέτει και την κατάλληλη παραμετροποίηση, ώστε η εφαρμογή να γνωρίζει που ακριβώς θα πρέπει να κάνει commit τα αρχεία, κάτι το οποίο έχει οριστεί απευθείας στον κώδικα, και δεν χρειάζεται να φροντίσει ο χρήστης για αυτό. Σημαντική βοήθεια για την τήρηση καλύτερου ιστορικού εκδόσεων του κώδικα αποτελεί η δυνατότητα προσθήκης σχολίου σε κάθε commit, ώστε εάν ο χρήστης χρειαστεί να ανατρέξει στο ιστορικό να βοηθηθεί από αυτά και να γνωρίζει ακριβώς τι άλλαξε ή προστέθηκε σε κάθε commit, revert κλπ.

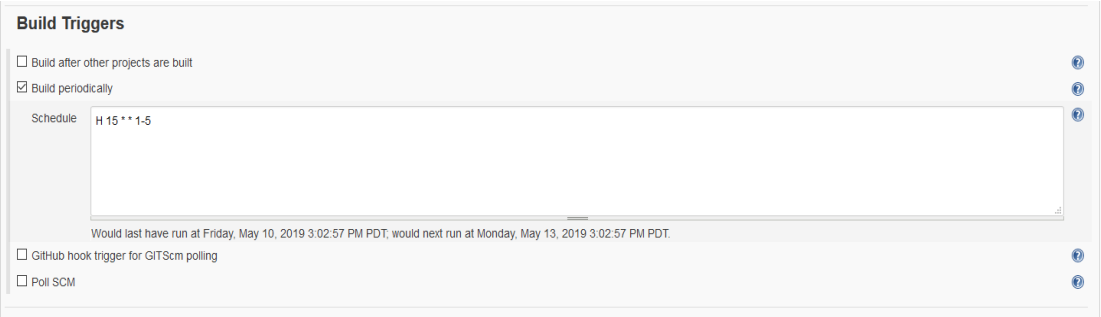
Ο προγραμματιστής αρκεί να γράψει το σχόλιο ή την περιγραφή του commit δίπλα στην ετικέτα “Log Text”.

Βήμα 6. Μεταφόρτωση αλλαγών στον server

Το τελευταίο βήμα της εφαρμογής έχει ως σκοπό την αντιγραφή των αλλαγών του κώδικα σε τοποθεσία του δικτύου στην οποία έχει πρόσβαση και ο Release Server. Αυτό το μονοπάτι αρχείων φαίνεται δίπλα στην ετικέτα “Path On Server”. Η πηγή των αρχείων υπάρχει πιο πάνω στα κάτω από τις δύο ετικέτες (“Back End” και “Front End”). Από εκεί η εφαρμογή θα διαβάσει τα αρχεία του κώδικα και θα τα αντιγράψει στον φάκελο προορισμού (“Path On Server”). Αυτό επιτυγχάνεται με μια σχετικά απλή υλοποίηση της C#, μέσω των εντολών File.Copy() για κάθε αρχείο που εντοπίζεται. Τα αντίστοιχα Progress Bars μας ενημερώνουν για την πρόοδο της διαδικασίας. Σε αυτό το σημείο έχει ολοκληρωθεί ένας κύκλος συνεχούς ενσωμάτωσης μέσω της εφαρμογής και απομένει το αυτοματοποιημένο σε ημερήσια βάση, έβδομο βήμα, το οποίο εκτελεί εντολές “Build”, μέσω του Jenkins στον Release Server.

Βήμα 7. Προγραμματισμένα builds στο Jenkins στον server

Για τις ανάγκες της αυτοματοποίησης, έχουν σχεδιαστεί και υλοποιηθεί ημερήσιες εργασίες στο Jenkins, οι οποίες εκτελούν builds τόσο στο front-end κομμάτι που έχει γίνει deploy, όσο και στο back-end. Αυτές τρέχουν μέσω Build Triggers δηλαδή χρονοπρογραμματισμένων εργασιών σε ημερήσια βάση από Δευτέρα ως Παρασκευή.



Εικόνα 17: Build triggers για αυτοματοποιημένη εκτέλεση εργασιών σε Jenkins

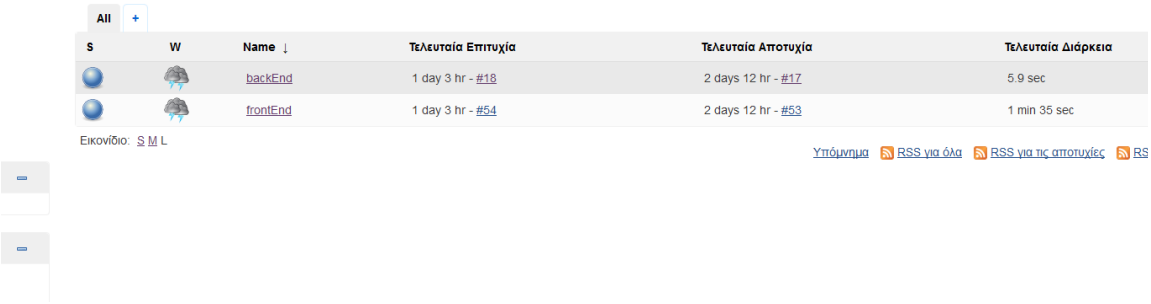
Για να επιτευχθεί build στο back-end αρχικά πηγαίνουμε στο φάκελο που υπάρχει ο κώδικας του και έπειτα εκτελούμε την εντολή dotnet build.

Εικόνα 18: Εντολές για build του .NET Core project στο Jenkins

Αντίστοιχα για το front-end πηγαίνουμε στο φάκελο του project και τρέχουμε την εντολή npm run ng build.

Εικόνα 19: Εντολές για build του Angular 6 project στο Jenkins

Στην αρχική οθόνη που καταγράφονται τα jobs μπορούμε να δούμε την κατάσταση του τελευταίου που εκτελέστηκε, πριν πόσες ώρες συνέβη αυτό, πόση ώρα χρειάστηκε κλπ.

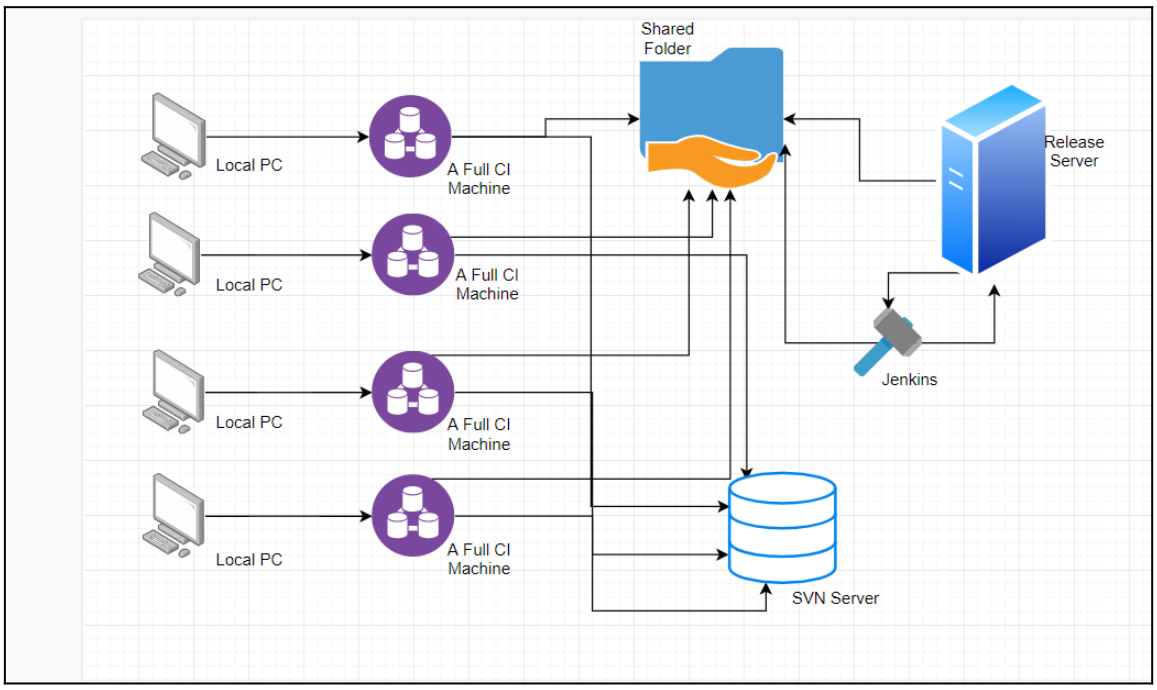


Εικόνα 20: Κατάσταση των δύο εργασιών στο Jenkins

4.2 Τεχνικές δυσκολίες – αστοχίες

4.2.1 Επικοινωνία τοπικού υπολογιστή με virtual machine

Η διαδικασία του deploy είναι αρκετά περίπλοκη και συχνά συναντά πολλές τεχνικές δυσκολίες. Αν για παράδειγμα ο Release Server τοποθετηθεί σε υπολογιστή με διεύθυνση εκτός δικτύου, τότε η πιο συμβατή λύση είναι η χρήση τεχνολογίας FTP (FileTransferProtocol – Πρωτόκολλο μεταφοράς αρχείων) η οποία απαιτεί προηγμένη υλοποίηση και πολλές φορές έχει αστοχίες στην εκτέλεση της όταν οι δυνατότητες του υλικού είναι περιορισμένες ή το σήμα δικτύου ασθενές.



Εικόνα 21: Η εφαρμογή A Full CI Machine σε ένα πλήρες προγραμματιστικό περιβάλλον

Επίσης είναι αρκετά δύσκολο για λόγους ασφαλείας να μεταφορτωθούν τα αρχεία σε άλλο υπολογιστή, ακόμα και με τη χρήση ελέγχου στοιχείων πρόσβασης. Έτσι, μια αρκετά συμβατή λύση είναι η δημιουργία κοινόχρηστων φακέλων δικτύου, στους

οποίους θα έχουν πρόσβαση όλοι οι τοπικοί υπολογιστές αλλά και ο Release Server. Ωστόσο, από τη στιγμή που οι φάκελοι αυτοί είναι πολλαπλά προσβάσιμοι, αυξάνεται παράλληλα και το ενδεχόμενο κινδύνων ασφαλείας.

4.2.2 Συμβατότητα Angular με Jenkins

Το Jenkins ως δημοφιλές εργαλείο συνεχούς ενσωμάτωσης, προσπαθεί να είναι συμβατό με όσες το δυνατόν περισσότερες τεχνολογίες είναι εφικτό. Για αυτό το σκοπό αναπτύσσονται διάφορα πρόσθετα (plugins) ώστε να απλοποιηθεί σημαντικά το build, deploy κλπ για κάθε τεχνολογία. Ωστόσο, ακόμη και σήμερα δεν έχει αναπτυχθεί κάποιο αξιόλογο πρόσθετο για τεχνολογίες Angular. Έτσι ο χρήστης θα αναγκαστεί να προσφύγει απευθείας σε εντολές τις οποίες θα χρησιμοποιούσε απευθείας μέσω της γραμμής εντολών (cmd). Στην προκειμένη για να προχωρήσει σε build θα τρέξει την εντολή ng build της Angular.

4.2.3 Δυσκολία συγχρονισμού SVN με Release Server

Μια επίσης σημαντική δυσκολία ήταν αυτή του συγχρονισμού ανάμεσα στο εργαλείο εκδόσεων λογισμικού και του server όπου θα τρέχει η εφαρμογή. Για λόγους δικαιωμάτων και μη επαρκούς εμπειρίας στο SVN, δεν έγινε εφικτή η πρόσβαση των τοπικών υπολογιστών αλλά και του server ταυτόχρονα στο versioning, με αποτέλεσμα το SVN να στηθεί μόνο τοπικά και το deploy να μην είναι αυτοματοποιημένα συγχρονισμένο με τα διάφορα commits. Βέβαια η εφαρμογή A Full CI Machine παρέχει αυτή τη δυνατότητα, αλλά σε διαφορετικά βήματα. Το ιδανικότερο θα ήταν να υπάρχουν διαφορετικά branches για κάθε επίπεδο παραγωγής (local, release).

5 Επίλογος

Η παρούσα διπλωματική μελέτησε εργαλεία Continuous Integration και προχώρησε στη χρήση, εγκατάσταση και ανάπτυξη του καταλληλότερου (Jenkins) για τα συγκεκριμένα δεδομένα. Επίσης ανέπτυξε μια διαδικτυακή εφαρμογή με χρήση τελευταίων τεχνολογιών ανάπτυξης λογισμικού και βάσεων δεδομένων. Τέλος δημιούργησε ένα μοντέρνο, αρκετά περιεκτικό σε δυνατότητες εργαλείο συνεχούς ενσωμάτωσης το οποίο αλληλεπιδρά με μεγάλη επιτυχία με τη διαδικτυακή εφαρμογή και προετοιμάζει το λογισμικό κατάλληλα πριν φθάσει στο σημείο όπου επεξεργάζεται από το Jenkins.

Όλο το παραπάνω αποτελεί μια ολοκληρωμένη και αρκετά ρεαλιστική υλοποίηση ενός κύκλου Continuous Integration σε μια σύγχρονη εφαρμογή, με πολλές ωστόσο δυνατότητες για επεκτάσεις και βελτιώσεις.

5.1 Σύνοψη και συμπεράσματα

Βασικό συμπέρασμα από τη μελέτη των εργαλείων Continuous Integration και ανάπτυξη νέων αποτελεί η ένδειξη πως η γενικότερη φιλοσοφία τους μπορεί να είναι παρόμοια, ωστόσο ο συνδυασμός των βημάτων, των σεναρίων και των τεχνολογιών που χρησιμοποιούνται μπορεί να ποικίλει και να οδηγήσει σε μοναδικές υλοποιήσεις για αρκετά σπάνιους συνδυασμούς δεδομένων. Ο βασικός ωστόσο πυλώνας μια τέτοιας εφαρμογής αποτελείται από βασικές έννοιες του CI, όπως build, commit,run tests, deploy κ.α.

5.2 Όρια και περιορισμοί της έρευνας

Η παρούσα διπλωματική εργασία και οι αντίστοιχες δύο εφαρμογές που αναπτύχθηκαν για τις ανάγκες της καλύπτουν ένα αρκετά ρεαλιστικό και σύνηθες σενάριο Continuous Integration, ωστόσο δεν μπορούν σε καμία περίπτωση να αντικαταστήσουν τα υπάρχοντα εργαλεία, όπως Jenkins, TeamCity κλπ.

Η εφαρμογή A Full CI Machine, μπορεί να καλύψει μόνο .NET projects, MySQL ως βάση δεδομένων και γενικότερα, μόνο τις τεχνολογίες στις οποίες είναι ανεπτυγμένη η εφαρμογή Travel Stories. Χρειάζεται πολύ μεγαλύτερη υλοποίηση, ίσως και επανασχεδιασμός ώστε η εφαρμογή συνεχούς ενσωμάτωσης να υποστηρίζει περισσότερες τεχνολογίες και σενάρια συνδυασμών αυτών.

5.3 Μελλοντικές Επεκτάσεις

Η συγκεκριμένη εργασία μέσω της ανάλυσης τεχνικών συνεχούς ολοκλήρωσης και της ανάπτυξης δύο έργων λογισμικού με σκοπό την εφαρμογή των παραπάνω τεχνικών καλύπτει ένα σημαντικό φάσμα της έννοιας της συνεχούς ενσωμάτωσης τόσο σε θεωρητικό, όσο και σε προγραμματιστικό επίπεδο.

Ωστόσο, μπορούν να προστεθούν κάποιες επεκτάσεις στην εφαρμογή για διασφάλιση μεγαλύτερης πληρότητας ελέγχων και προσέγγισης, όσο το δυνατόν περισσότερο ενός πραγματικού έργου λογισμικού στη βιομηχανία.

Αρχικά πριν το στάδιο της αντιγραφής των βελτιωμένων αρχείων στον server, υπάρχει δυνατότητα τα αρχεία να αντιγραφούν και να εγκατασταθούν(deploy) στον τοπικό server (IIS) ώστε η εφαρμογή Travel Stories να γίνει προσβάσιμη σε τοπικό περιβάλλον χωρίς τη ανάγκη χρήσης του περιβάλλοντος ανάπτυξης κώδικα visual studio. Αυτό μπορεί να επιτευχθεί με κατάλληλη παραμετροποίηση στον IIS και παράλληλα κλήση της λειτουργίας deploy πάνω στην εφαρμογή Travel Stories.

Επιπλέον η εφαρμογή A Full CI Machine μπορεί να έχει μια έξτρα φόρμα παραμετροποίησης σε θέματα που αφορούν τον server πάνω στον οποίο θα ενσωματωθούν τα αρχεία στο τελευταίο βήμα. Αυτές οι ρυθμίσεις μέσω γραμμής εντολών θα περνούν στον server και θα γίνονται ακόμα πιο αυτοματοποιημένες οι λειτουργίες και σε αυτό το επίπεδο.

Μία ακόμη σημαντική επέκταση θα μπορούσε να είναι η δημιουργία μια αναφοράς στο τέλος κάθε κύκλου ενσωμάτωσης στην εφαρμογή A Full CI Machine. Αυτή θα περιλαμβάνει καταγραφή των αλλαγών ανά αρχείο, περιγραφικό σχόλιο για κάθε commit, χρόνους build για front end και back end της εφαρμογής Travel Stories, καθώς και λάθη τα οποία εντοπίστηκαν και ένα γενικότερο status το οποίο θα προκύπτει από όλα τα παραπάνω.

Τέλος η web εφαρμογή θα μπορούσε να εμπλουτιστεί με νέες λειτουργίες ώστε να φτάσει σε ένα ακόμα υψηλότερο επίπεδο χρησιμότητας και λειτουργικότητας. Θα μπορούσε να κρατάει session πάνω στο login το οποίο θα λήγει μετά από ένα συγκεκριμένο χρονικό όριο, να έχει δυνατότητα logout, χρήση τεχνολογίας recaptcha μετά από συγκεκριμένο ανεπιτυχή αριθμό απόπειρας εισόδου στην εφαρμογή καθώς και

καταγραφής cookies για βελτιστοποίηση. Μπορεί επίσης να επεκτείνει τις λειτουργίες της μέσω της διαγραφής και επεξεργασίας ιστορίας, τη προσθήκη βίντεο, αλλά και δυνατότητα κοινοποίησής της σε μέσα κοινωνικής δικτύωσης.

Βιβλιογραφία

1. **Martin Fowler, *Continuous Integration*, published online 1 May 2006, http://www.dccia.ua.es/dccia/inf/asignaturas/MADS/2013-14/lecturas/10_Fowler_Continuous_Integration.pdf**

2. **Ade Miller, *A Hundred Days of Continuous Integration*, Micosoft Corporation, http://www.ademiller.com/tech/reports/a_hundred_days_of_continuous_integration.pdf**

3. **Ιώαννης Τσαφαράς, *Ανάπτυξη και Σχεδίαση Αυτοματοποιημένης Υποδομής Continuous Integration σε έργα λογισμικού*, Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, Οκτώβριος 2017, http://ikee.lib.auth.gr/record/297475/files/Thesis_Ioannis_Tsafaras_Continuous_Integration.pdf**

4. **https://www.softwaretestinghelp.com/tools/top-40-static-code-analysis-tools/**

5. **https://www.upguard.com/articles/teamcity-vs.-jenkins-for-continuous-integration**

6. **https://daveaglick.com/posts/running-a-design-time-build-with-msbuild-apis**

7. **https://www.amarinfotech.com/difference-between-net-core-2-0-vs-net-framework.html**

8. **https://www.guru99.com/net-framework.html**

9. **https://devblogs.microsoft.com/dotnet/introducing-net-5/?fbclid=IwAR2P8XMAsfpt_ah4ly-cOPnxc9yr6rOv3D2FL4O1lbMsHDeI57TBMciPqu0**

10. **https://lifehacker.com/five-best-virtual-machine-applications-5714966**

11. **https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems**

12. **https://www.codeinwp.com/blog/angular-vs-vue-vs-react/**

13. **https://www.sonarqube.org/features/clean-code/**

14. **https://jenkins.io/doc/**

15. **https://www.thoughtworks.com/continuous-integration**

16. **https://medium.com/@levifuller/building-an-angular-application-with-asp-net-core-in-visual-studio-2017-visualized-f4b163830eaa**