ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΑΓΡΟΝΟΜΩΝ ΚΑΙ ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ

ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΓΕΩΠΛΗΡΟΦΟΡΙΚΗ»

Διαχείριση Χωρικών Βάσεων Δεδομένων για την Παρακολούθηση της Εναέριας Κυκλοφορίας

ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του

ΑΛΕΞΑΝΔΡΟΥ ΓΚΙΖΛΗ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2012

.....

ΑΛΕΞΑΝΔΡΟΣ ΓΚΙΖΛΗΣ

Αξιωματικός ΠΑ, Μηχανικός - Ειδικότητα Τηλεπικοινωνιών - Ηλεκτρονικών

Περίληψη

Ο σκοπός της εργασίας είναι η ανάπτυξη μιας εφαρμογής για την λήψη, αποθήκευση και προβολή ιχνών πολιτικών αεροσκαφών στην Πολεμική Αεροπορία (ΠΑ). Καθώς δεν είναι διαθέσιμα πραγματικά δεδομένα, η λήψη θα γίνεται από έναν εξομοιωτή - γεννήτρια ιχνών. Η αποθήκευση θα γίνεται σε μια γεωχωρική βάση δεδομένων με σκοπό τη δυνατότητα μελλοντικής επιχειρησιακής αξιοποίησης των δεδομένων. Η προβολή θα είναι περιορισμένη σε αεροπορική εικόνα πραγματικού χρόνου, αλλά και κάποιας ορισμένης στιγμής στο παρελθόν.

Η ΠΑ διαθέτει ένα πλήθος radar για την ανίχνευση αεροσκαφών στον εναέριο χώρο της Ελλάδας, που συνοδεύεται από το ανάλογο λογισμικό για την παρουσίαση των δεδομένων που συλλέγονται. Στα κέντρα επιχειρήσεων όμως είναι εξαιρετικά χρήσιμο να υπάρχει διαθέσιμη και η αεροπορική εικόνα που διαθέτει η Υπηρεσία Πολιτικής Αεροπορίας (ΥΠΑ), και κατά συνέπεια το ευρύ κοινό, για επιχειρησιακούς σκοπούς. Το λογισμικό που χρησιμοποιείται αυτή τη στιγμή για το σκοπό αυτό, δεν έχει τη δυνατότητα αποθήκευσης των ιχνών. Το κενό αυτό θα προσπαθήσει να καλύψει η παρούσα εργασία.

Λέξεις Κλειδιά: <<Αεροπορική εικόνα, αεροσκάφος, ίχνη αεροσκαφών, γεωχωρική βάση δεδομένων, Πολεμική Αεροπορία>>

Abstract

The scope of this thesis is the development of an application that will receive, store and present civil aviation aircraft tracks for the Hellenic Air Force (HAF). Since real data is not available, the application will receive tracks from a simulator - generator of aircraft tracks. The tracks will be stored in a geospatial database so they can be used in the future by HAF. The presentation will be limited in displaying real and past time data to the user, with no further analysis.

HAF has a number of radars tracking aircraft in the Greek airspace, accompanied by software capable of presenting the data collected. In operations centers though, is it invaluable to have available the airspace status as the Hellenic Civil Aviation Authority (HCAA) sees it, which is how the general public sees it. The software currently being used for that purpose does not have the capability of storing the data collected in any format. It is exactly this capability that the development of this thesis' application will attempt to implement.

Keywords: <<Airspace, aircraft, aircraft track, geospatial database, Hellenic Air Force>>

Πίνακας περιεχομένων

1 Εισαγωγή 1

1.1 Αεροπορική εικόνα 1

1.2 Αντικείμενο εργασίας 2

1.3 Οργάνωση κειμένου 4

2 Σχετικές εργασίες 6

2.1 Διαχείριση στόλου οχημάτων 6

2.2 PALLAS client 7

2.3 Αποθήκευση μεγάλου όγκου πληροφοριών 7

3 Ανάλυση Απαιτήσεων Συστήματος 8

3.1 Αρχιτεκτονική 8

3.2 Περιγραφή Λειτουργιών 9

3.2.1 Γεννήτρια Ιχνών 9

3.2.2 Μονάδα Εισόδου 10

3.2.3 Μονάδα Αποθήκευσης 11

3.2.4 Μονάδα Παρουσίασης 12

3.3 Μοντέλο Οντοτήτων Συσχετίσεων 13

4 Πλατφόρμες, εργαλεία, πρότυπα 14

4.1 Microsoft SQL Server 2008 R2 Express 14

4.2 Web Feature Service 15

4.3 GeoServer 15

4.4 GeoTools 15

4.5 NetBeans 16

4.6 Δεδομένα και βιβλιοθήκες 17

5 Σχεδίαση Συστήματος 18

5.1 Αρχιτεκτονική 18

5.2 Υποσυστήματα 19

5.2.1 Γεννήτρια Ιχνών 19

5.2.2 Μονάδα Εισόδου *20*
5.2.3 Μονάδα Αποθήκευσης *22*
5.2.4 Μονάδα Παρουσίασης *23*
5.3 Modules 23
5.3.1 Track *24*
5.3.2 GeoTools *25*
5.3.3 Global Settings *26*
5.3.4 Global Lookup *26*
5.3.5 Track generator beans *27*
5.3.6 Track generator settings *28*
5.3.7 Generator module *28*
5.3.8 Track generator status *34*
5.3.9 Track receiver beans *34*
5.3.10 Track receiver *34*
5.3.11 Track receiver window *35*
5.3.12 Track WFS sender beans *36*
5.3.13 Track WFS sender settings *37*
5.3.14 Track WFS sender *37*
5.3.15 Track WFS sender status *38*
5.3.16 Track client WFS receiver *39*
5.3.17 JCalendarWrapper *41*
5.3.18 Track client map *41*
5.4 Βάση Δεδομένων 46
5.4.1 Δομή βάσης *46*
5.4.2 Ερωτήματα *47*
6 Υλοποίηση 48
6.1 Λεπτομέρειες υλοποίησης 48
6.1.1 Δημιουργία ενός course *48*
6.1.2 Δημιουργία WFS insert request *49*
6.1.3 Ερώτημα SQL για τα τελευταία διαθέσιμα ίχνη *52*
6.1.4 Διαδικασία σχεδίασης ενός ίχνους *53*

6.2 Εγκατάσταση 55

7 Έλεγχος 57

7.1 Μεθοδολογία ελέγχου 57

7.2 Αναλυτική παρουσίαση ελέγχου 58

7.2.1 Γεννήτρια ιχνών 58

7.2.2 Δέκτης Μονάδας Εισόδου 58

7.2.3 Αποστολέας Μονάδας Εισόδου 59

7.2.4 Μονάδα Αποθήκευσης 60

7.2.5 Μονάδα Παρουσίασης 60

8 Επίλογος 64

8.1 Σύνοψη και συμπεράσματα 64

8.1.1 Θετικά σημεία 64

8.1.2 Αρνητικά σημεία 65

8.1.3 Συμπεράσματα 65

8.2 Μελλοντικές επεκτάσεις 66

9 Βιβλιογραφία 68

1

Εισαγωγή

1.1 Αεροπορική εικόνα

Η αεροπορική εικόνα είναι ένα σύνολο πληροφοριών που απεικονίζονται σε ένα χάρτη και αφορούν αεροσκάφη. Υπάρχουν δύο τύποι αεροπορικής εικόνας, η πολιτική και η στρατιωτική. Η διάκριση γίνεται λόγω της προέλευσης της πληροφορίας, του χώρου χρήσης της, της διαβάθμισής της, της ακρίβειάς της, αλλά και του περιεχομένου. Στην πρώτη περίπτωση η πληροφορία προέρχεται από τα radar της Πολεμικής Αεροπορίας, χρησιμοποιείται σε στρατιωτικούς χώρους, έχει τη μέγιστη διαβάθμιση και καλύτερη ακρίβεια, ενώ στη δεύτερη περίπτωση η πληροφορία προέρχεται κυρίως από τα radar των αεροδρομίων, χρησιμοποιείται από πλήθος εργαζομένων στην ΥΠΑ, διατίθεται δημόσια, και έχει χειρότερη ακρίβεια. Η εργασία αυτή ασχολείται αποκλειστικά με την πολιτική αεροπορική εικόνα.

Στην Ελλάδα, η αεροπορική εικόνα παράγεται από το Σύστημα Επεξεργασίας Σχεδίων Πτήσεως & Δεδομένων Radar - Phased Hellenic Air Traffic Control System (PALLAS). Τα δεδομένα συλλέγονται από ένα πλήθος radar ανά την ελληνική επικράτεια, σε έναν υπολογιστή που ονομάζεται Radar Front Processor (RFPS). Τα πιο γρήγορα από τα radar που υπάρχουν, εκτελούν μία περιστροφή ανά 5 δευτερόλεπτα, τα περισσότερα ανά 8. Στην καλύτερη περίπτωση, αυτός είναι και ο ρυθμός αποστολής πληροφοριών στον RFPS. Στη συνέχεια προωθούνται στον Radar Data Processor (RDPS) όπου και γίνεται η συσχέτιση των

ιχνών που έχουν ληφθεί από παραπάνω από ένα radar. Έτσι δημιουργείται η πληροφορία η οποία διατίθεται μέσω δικτύου σε clients όπου απαιτείται. Χωρίς να λαμβάνουμε υπόψη ακραίες περιπτώσεις, είναι ασφαλές να θεωρήσουμε ότι σε ώρες αιχμής βρίσκονται στον εναέριο χώρο περίπου 300 αεροσκάφη. Άρα ο client λαμβάνει περίπου 300 ίχνη ανά 5 δευτερόλεπτα.

Η ΠΑ είναι συνδεδεμένη με το PALLAS, λαμβάνει την αεροπορική εικόνα και την διανέμει εσωτερικά σε όσα τερματικά απαιτείται. Τα τερματικά και το λογισμικό που χρησιμοποιείται είναι της ΥΠΑ. Σύμφωνα με τη σημερινή μορφή του συστήματος, ο τελικός χρήστης έχει τη δυνατότητα να βλέπει που βρίσκονται αεροσκάφη στον εναέριο χώρο της Ελλάδας, έχει διαθέσιμες διάφορες πληροφορίες για αυτά, αλλά και πλήθος άλλων πληροφοριών, όπως είναι τα σχέδια πτήσεων, τα όρια των FIR, προειδοποιήσεις για ενδεχόμενες παραβιάσεις ορίων, και άλλες.

Η απαίτηση που έχει προκύψει από την ΠΑ για επιχειρησιακούς σκοπούς και δεν καλύπτεται από το PALLAS είναι η αποθήκευση των ιχνών των αεροσκαφών ώστε να μπορούν να αξιοποιηθούν μελλοντικά. Έτσι, αν ο χρήστης επιθυμεί να δει την πορεία ενός αεροσκάφους από την απογείωση μέχρι την προσγείωση, δεν έχει τη δυνατότητα. Επίσης δε μπορεί να δει την αεροπορική εικόνα όπως ήταν μερικές μέρες πριν, δε μπορεί να δει ποια ήταν η ελάχιστη απόσταση που είχαν δύο αεροσκάφη σε όλη τη διάρκεια της πτήσης τους, και γενικότερα δε μπορεί εκ των υστέρων να έχει πρόσβαση σε πληροφορία πέραν της εικόνας πραγματικού χρόνου που του παρέχεται.

1.2 Αντικείμενο εργασίας

Η εργασία έχει στόχο τη δημιουργία μιας εφαρμογής που θα μπορεί να λαμβάνει πληροφορίες ιχνών αεροσκαφών από διάφορες πηγές, να τις αποθηκεύει σε μια γεωχωρική βάση δεδομένων, και να τις προβάλει στον τελικό χρήστη.

Η εφαρμογή θα μπορούσε να περιοριστεί στην επεξεργασία ιχνών που θα λαμβάνει από το PALLAS. Καθώς όμως το πρωτόκολλο επικοινωνίας του PALLAS δεν είναι ευρέως διαθέσιμο, και δε θα μπορούσε να δημοσιευτεί σε μια εργασία προσβάσιμη από οποιονδήποτε, επιλέχτηκε η δεύτερη καλύτερη λύση. Η εφαρμογή θα μπορεί με την προσθήκη modules (ή extensions, ή plug-ins, ή επεκτάσεων όπως αλλιώς λέγονται) να συνδέεται σε οποιαδήποτε πηγή πληροφοριών ιχνών. Τα ίχνη θα συγκεντρώνονται από ένα module, θα μετατρέπονται σε μια ειδική εσωτερική μορφή, και θα αποστέλλονται σε ένα εξυπηρετητή για αποθήκευση σε μια βάση δεδομένων. Θα υλοποιηθεί μάλιστα ένα module που θα παράγει ίχνη, για σκοπούς ελέγχου και προσομοίωσης.

Η εξυπηρετητής θα έχει δύο ρόλους. Ο πρώτος θα είναι η αποθήκευση στη γεωχωρική βάση δεδομένων όλων των ιχνών που αποστέλλονται. Ο δεύτερος θα είναι η αποστολή δεδομένων από τη γεωχωρική βάση στους τελικούς χρήστες. Εδώ ενδέχεται να υπάρξει πρόβλημα στην ταχύτητα απόκρισης, καθώς ο ρυθμός εισαγωγής και ανάγνωσης δεδομένων αναμένεται να είναι μεγάλος. Σίγουρα δε θα μπορεί να εξυπηρετηθεί μεγάλος αριθμός χρηστών, αλλά ούτε και απαιτείται. Τα τερματικά PALLAS που είναι ήδη εγκατεστημένα στην ΠΑ έχουν μονοψήφιο αριθμό. Η εφαρμογή όμως θα πρέπει να διαθέτει ένα μηχανισμό κατανομής του φόρτου ώστε να υπάρχει η δυνατότητα εξυπηρέτησης μεγαλύτερου αριθμού χρηστών.

Η εφαρμογή του τελικού χρήστη θα μπορεί να προβάλλει την αεροπορική εικόνα σε «πραγματικό» χρόνο, αλλά και στιγμιότυπα από το παρελθόν, και στις δύο περιπτώσεις με προαιρετικά φίλτρα που θα δίνει. Ένα στοιχειώδες υπόβαθρο θα χρησιμοποιείται στην οπτικοποίηση, καθώς και συμβολισμοί που θα προσομοιάζουν το PALLAS ώστε να είναι οικείο στους χρήστες.

Κάνοντας όλα τα παραπάνω, καλύπτει εν μέρει την απαίτηση αποθήκευσης των ιχνών, όπως αυτή περιγράφτηκε στο τέλος της παραγράφου 1.1. Η πλήρης αντικατάσταση του λογισμικού τελικού χρήστη του PALLAS από αυτή την εφαρμογή θα απαιτούσε πληθώρα επιπλέον χαρακτηριστικών και θα μπορούσε να αποτελέσει αντικείμενο απασχόλησης για μια ομάδα εργασίας για μεγάλο χρονικό διάστημα. Είναι επομένως πέρα από τα όρια της εργασίας αυτής. Παρόλα αυτά, η εργασία αυτή θα υλοποιήσει τις βασικές λειτουργίες και θα υλοποιήσει ένα υπόβαθρο όπου σταδιακά θα μπορούσε κανείς προσθέτοντας modules να το βελτιώσει όσο επιθυμεί. Θα είναι δηλαδή εύκολα επεκτάσιμη.

Όλα τα παραπάνω πρέπει να γίνουν λαμβάνοντας υπόψη ορισμένους περιορισμούς:

1. Θα πρέπει να τρέχει στο εσωτερικό κλειστό και ασφαλές δίκτυο της ΠΑ, χωρίς καμία επαφή με το διαδίκτυο. Αυτό αποκλείει τη χρήση λογισμικού απεικόνισης γεωχωρικής πληροφορίας που αντλεί πληροφορίες για το υπόβαθρο από το διαδίκτυο.
2. Θα πρέπει να κοστίζει όσο το δυνατόν λιγότερο. Η χρήση ανοιχτού λογισμικού είναι η λύση σε αυτό τον περιορισμό.
3. Θα πρέπει να χρησιμοποιεί ως λειτουργικό σύστημα τα Microsoft Windows και ως γεωχωρική βάση δεδομένων την Microsoft SQL Server, καθώς η ΠΑ διαθέτει ήδη τις απαραίτητες άδειες χρήσης και για λόγους εκπαίδευσης, συντήρησης, ευκολίας και συμβατότητας αποτελεί μονόδρομο.
4. Θα πρέπει να βασίζεται σε αναγνωρισμένα και δοκιμασμένα πρότυπα, βιβλιοθήκες, και λογισμικό όπου γίνεται, ώστε να είναι εύκολη η μελλοντική υποστήριξη, επέκταση, συντήρηση και εκμάθηση της εφαρμογής. Η ΠΑ έχει πολλές φορές

αντιμετωπίσει πρόβλημα με λογισμικό μη-προτυποποιημένο με ελλειπή τεκμηρίωση. Επιπλέον αυτός ο περιορισμός θα βοηθήσει στη σταθερότητα του συστήματος.

Η συνεισφορά της εργασίας συνοψίζεται ως εξής:

1. Η εφαρμογή θα πρέπει να έχει τη δυνατότητα επέκτασης ώστε να συνδέεται με πολλά και διαφορετικά συστήματα παροχής ιχνών αεροσκαφών. Προκειμένου να γίνει αυτό, θα πρέπει να υποστηρίζει πληροφορίες ιχνών διαφορετικής μορφής, ακόμα και άγνωστης μορφής. Μελετήθηκαν τρία πρωτόκολλα μετάδοσης πληροφοριών ιχνών, και εσωτερικά υλοποιήθηκε μια μορφή ίχνους που υπερκαλύπτει και τα τρία. Για την ταυτόχρονη κάλυψη πολλών πηγών, προστέθηκε ένα επίπεδο αφαίρεσης. Αντί δηλαδή η πηγή να στέλνει τα δεδομένα στον server απευθείας, τα στέλνει σε ένα ενδιάμεσο σταθμό, όπου μετατρέπονται στην εσωτερική μορφή. Αυτός ο σταθμός στέλνει στη συνέχεια τα «ομογενοποιημένα» δεδομένα στον server. Αν απαιτηθεί στο μέλλον, θα μπορούσε επιπλέον να επεξεργάζεται τα δεδομένα πριν τα στείλει, ή να τα στέλνει σε πολλούς server ταυτόχρονα. Η modular δομή της εφαρμογής επιτρέπει την εύκολη αντικατάσταση ή προσθήκη οποιουδήποτε module, κάνοντάς τη πραγματικά ευέλικτη.
2. Χρησιμοποιήθηκε ανοιχτό λογισμικό σε όλα τα επίπεδα, εκτός από το λειτουργικό σύστημα και τη βάση δεδομένων, όπου βάση των προδιαγραφών είναι συγκεκριμένα και δε μπορούσαμε να επέμβουμε. Το κόστος προμήθειας λογισμικού για την ΠΑ είναι πρακτικά ανύπαρκτο.
3. Αναπτύχθηκε μια στοιχειώδης γεννήτρια ιχνών αεροσκαφών, με δυνατότητα ρύθμισης του πλήθους και του ρυθμού παραγωγής ιχνών. Αυτό κατέστησε δυνατή τη μελέτη της συμπεριφοράς της εφαρμογής κάτω από διαφορετικές συνθήκες, και την εξαγωγή συμπερασμάτων.
4. Δοκιμάστηκε ένα σύνολο εργαλείων λογισμικού σε μια εφαρμογή με αυξημένες απαιτήσεις και αναδείχτηκαν τα πλεονεκτήματα και τα μειονεκτήματά τους.

1.3 Οργάνωση κειμένου

Στο κεφάλαιο 1 έγινε μια εισαγωγή στο πρόβλημα και τη λύση που αποτελεί το αντικείμενο αυτής της εργασίας. Στο κεφάλαιο 2 γίνεται αναφορά σε εργασίες σχετικές με τη μεταπτυχιακή εργασία και πώς αυτές την επηρέασαν. Στο κεφάλαιο 3 παρουσιάζονται συνοπτικά ορισμένα πρωτόκολλα, τεχνικές και μοντέλα που χρησιμοποιήθηκαν κατά την ανάπτυξη της εφαρμογής. Στο κεφάλαιο 4 αναλύονται οι απαιτήσεις του συστήματος και βάση αυτών προκύπτει η αρχιτεκτονική του. Στο κεφάλαιο 5 παρουσιάζεται η δομή του συστήματος όπως αυτή προκύπτει με βάση τις απαιτήσεις του κεφαλαίου 4. Στο κεφάλαιο 6

βλέπουμε ορισμένα σημαντικά ή δύσκολα σημεία της υλοποίησης αναλυτικά, καθώς και τα εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση. Στο κεφάλαιο 7 γίνεται η αξιολόγηση της εφαρμογής. Στο κεφάλαιο 8 αναλύονται οι δυνατότητες επέκτασης της εφαρμογής. Στο κεφάλαιο 9 παρουσιάζονται συνοπτικά τα συμπεράσματα της εργασίας. Στο τελευταίο κεφάλαιο παρατίθεται η βιβλιογραφία που χρησιμοποιήθηκε.

2

Σχετικές εργασίες

Η εργασία αυτή έχει πολλά κοινά στοιχεία με τη διαχείριση στόλου οχημάτων. Θα μπορούσε να πει κανείς μάλιστα ότι οι διαφορές είναι μικρές. Η πραγματικότητα διαφέρει, και οι λόγοι παρουσιάζονται στην παράγραφο 2.1.

Μια εργασία που κινείται σε παρεμφερή πλαίσια είναι η (Δρακόπουλος & Παναγιώτου, 2002) στην οποία γίνεται μια προσπάθεια δημιουργίας ενός client PALLAS. Παρουσιάζεται αναλυτικότερα στην παράγραφο 2.2.

Όσον αφορά την αποθήκευση πληροφοριών αισθητήρων μεγάλου όγκου ή/και ρυθμού, μια σχετική εργασία είναι η (Malensek et al. 2012) και την βλέπουμε στην παράγραφο 2.3.

2.1 Διαχείριση στόλου οχημάτων

Οι εφαρμογές διαχείρισης στόλου οχημάτων μπορούμε να πούμε ότι έχουν (στην πλειοψηφία τους) δύο σκέλη. Το ένα αφορά τη μη-χωρική διαχείριση των οχημάτων, που αφορά την καταγραφή των χαρακτηριστικών τους, την καταγραφή των οδηγών τους, την ενημέρωση για συντήρησή τους κ.α., και το δεύτερο σκέλος αφορά τη χωρική διαχείρισή τους. Ενδεικτικά αναφέρονται οι (TomTom) και (iLink, 2010).

Στο πρώτο σκέλος αυτή η εργασία δεν έχει κανένα κοινό στοιχείο. Το PALLAS καλύπτει τις αντίστοιχες απαιτήσεις σε αυτό τον τομέα. Στο δεύτερο σκέλος υπάρχουν δύο διαφορές. Η μία είναι ότι οι εφαρμογές διαχείρισης στόλου οχημάτων δεν αποθηκεύουν τα στοιχεία που

συλλέγονται από του αισθητήρες αυτούσια. Υπόκεινται συνήθως σε μια διαδικασία που περιγράφεται στο (Patroumpas, Sellis 2004), κατά την οποία αποθηκεύονται στατιστικά του συνόλου των στοιχείων, και όχι τα ίδια τα στοιχεία. Η δεύτερη διαφορά είναι ότι η έμφαση δίνεται στην παρουσίαση των στοιχείων, ενώ στην εργασία αυτή η έμφαση δίνεται στη συλλογή και αποθήκευση.

2.2 PALLAS client

Στα πλαίσια της εργασίας που παραδίδουν οι Ίκαροι για την αποφοίτησή τους από τη Σχολή Ικάρων, δύο Ίκαροι εκπόνησαν την εργασία (Δρακόπουλος & Παναγιώτου, 2002), στην οποία επιχείρησαν να μελετήσουν το πρωτόκολλο επικοινωνίας του PALLAS και να αναπτύξουν μια εφαρμογή client για αυτό. Η εφαρμογή χειριζόταν τη γεωχωρική πληροφορία ως αριθμούς χωρίς γεωαναφορά, δεν αποθήκευε τα δεδομένα που λάμβανε, και το υπόβαθρο που χρησιμοποιούσε για την προβολή ήταν μία στατική εικόνα. Το σύστημα δεν ήταν επεκτάσιμο ή προσαρμόσιμο σε άλλες πηγές ιχνών, και γενικά διέφερε πολύ στους στόχους του από την παρούσα εργασία.

2.3 Αποθήκευση μεγάλου όγκου πληροφοριών

Ο λόγος που οι γεωχωρικές ροές δεδομένων από αισθητήρες συνήθως δεν αποθηκεύονται, είναι ο μεγάλος όγκος που απαιτείται για την αποθήκευση. Για να αντιμετωπιστεί αυτό το πρόβλημα εκπονήθηκαν μελέτες πάνω σε δύο κατευθύνσεις. Η μία ήδη αναφέρθηκε (Patroumpas, Sellis 2004) και αφορά την αποθήκευση στατιστικών στοιχείων ή μεταδεδομένων μόνο, και όχι των ίδιων των στοιχείων. Η άλλη κατεύθυνση είναι η ανάπτυξη συστημάτων που να μπορούν να ανταποκριθούν σε πολύ μεγάλη ροή δεδομένων, για την αποθήκευση και ανάκτησή τους. Μια μελέτη σε αυτή την κατεύθυνση είναι η (Malensek et al. 2012), στην οποία αναπτύχθηκε μια κατανεμημένη βάση δεδομένων που με ένα ειδικευμένο σύστημα αρχειοθέτησης και αλγόριθμους κατανομής, μοιράζει τα δεδομένα στους υπολογιστές μιας συστοιχίας. Μια υλοποίηση σαν και αυτή δεν είναι στα πλαίσια μείωσης του κόστους που τέθηκε ως απαίτηση σε αυτή την εργασία. Άλλωστε οι απαιτήσεις της εφαρμογής είναι τάξεις χαμηλότερες από αυτές που καλύπτει η παραπάνω μελέτη.

3

Ανάλυση Απαιτήσεων Συστήματος

Στα προηγούμενα κεφάλαια είδαμε ποιο είναι το αντικείμενο της εργασίας και ποιες παρεμφερείς εργασίες έχουν γίνει. Σε αυτό το κεφάλαιο θα δούμε πως διαμορφώνεται η αρχιτεκτονική του συστήματος ώστε να καλύπτει τις απαιτήσεις που τέθηκαν.

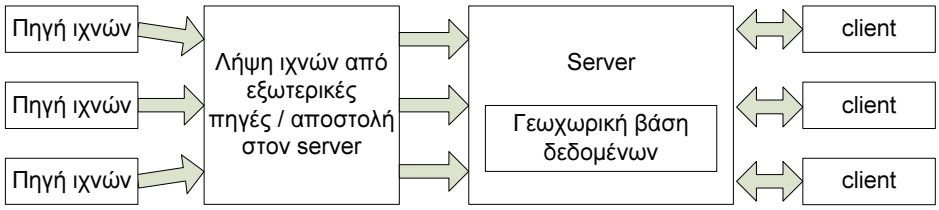
3.1 Αρχιτεκτονική

Η γενική μορφή της εφαρμογής είναι σχετικά απλή. Μπορεί αρχικά να χωριστεί σε τέσσερα μεγάλα επιμέρους τμήματα, όπου το καθένα εκτελεί μια γενική λειτουργία.

1. Το πρώτο τμήμα αφορά τις πηγές ιχνών. Σε ένα πραγματικό σενάριο, θα είχαμε πραγματικές πηγές δεδομένων. Εφόσον δεν έχουμε διαθέσιμες πραγματικές πηγές δεδομένων, και προκειμένου να είμαστε σε θέση να δοκιμάσουμε την εφαρμογή, είναι απαραίτητη η προσθήκη μιας εικονικής πηγής. Το πρώτο τμήμα λοιπόν αφορά την εικονική πηγή, ή εξομοιωτή ιχνών, ή γεννήτρια ιχνών που αναπτύχθηκε.
2. Το δεύτερο τμήμα αφορά τη λήψη των δεδομένων ιχνών από τις διαθέσιμες πηγές και την αποστολή τους σε ένα server προς αποθήκευση στη γεωχωρική βάση δεδομένων. Οι πηγές μπορεί να είναι πολλές, και κάθε πηγή εξυπηρετείται παράλληλα, δηλαδή τα δεδομένα της επεξεργάζονται και αποστέλλονται στον server ανεξάρτητα από τις υπόλοιπες πηγές.

3. Το τρίτο τμήμα αφορά τη λήψη των ιχνών από τον server, την αποθήκευσή τους, και την αποστολή ιχνών σε όσους clients θέλουν τα στοιχεία. Ο server μπορεί να λαμβάνει «κανονικοποιημένα» ίχνη από πολλές πηγές ταυτόχρονα, και να εξυπηρετεί πολλούς client ταυτόχρονα.
4. Το τέταρτο τμήμα είναι ο client, ο οποίος λαμβάνει τα στοιχεία που ζητάει από τον server, και τα προβάλλει στον τελικό χρήστη. Μπορούν να υπάρχουν πολλοί clients ενεργοί ταυτόχρονα.

Στο παρακάτω σχήμα απεικονίζεται η παραπάνω δομή.



Σχήμα 1: Γενική αρχιτεκτονική συστήματος

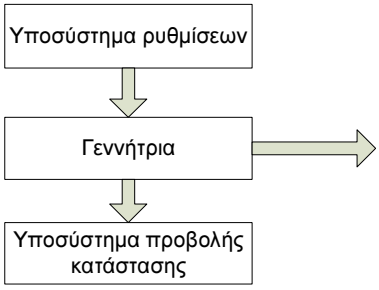
3.2 Περιγραφή Λειτουργιών

Σε αυτό το σημείο είναι σκόπιμο να ονοματίσουμε τα υποσυστήματα της εφαρμογής, προκειμένου να μη δημιουργηθεί σύγχυση όταν θα αναλυθούν περισσότερο.

Η πηγή ιχνών μπορεί να είναι το PALLAS, το Asterix (που παρουσιάζεται στο επόμενο κεφάλαιο), ένας εξωτερικός εξομοιωτής (Γεννήτρια Ιχνών θα ονομάσουμε την υλοποίηση της εργασίας) ή οποιοδήποτε άλλο σύστημα. Το υποσύστημα που λαμβάνει στοιχεία από τις πηγές, τα επεξεργάζεται και τα στέλνει στον server θα το ονομάσουμε Μονάδα Εισόδου. Το τρίτο υποσύστημα θα το ονομάσουμε Μονάδα Αποθήκευσης, ενώ το τέταρτο Μονάδα Παρουσίασης. Ας δούμε τα υποσυστήματα αναλυτικότερα.

3.2.1 Γεννήτρια Ιχνών

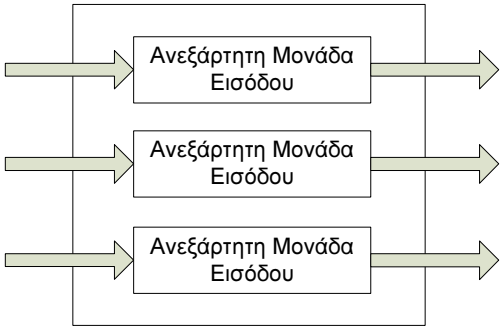
Η Γεννήτρια Ιχνών έχει αποκλειστικό στόχο να παράγει ίχνη που θα τροφοδοτούν την εφαρμογή προκειμένου να ελεγχθεί αποτελεσματικά κάτω από διάφορες συνθήκες. Επομένως, θέλουμε να παράγει ίχνη αεροσκαφών σχετικά ρεαλιστικά ώστε να έχουμε ένα οπτικό αποτέλεσμα εύκολα αντιληπτό, να μπορεί να εξομοιώσει πολλές πηγές ταυτόχρονα ώστε να μπορεί να διεξαχθεί ένα stress test, να παραμετροποιείται ως προς το ρυθμό και την ποσότητα των ιχνών που παράγει, και τέλος να προβάλει την κατάστασή της ώστε να έχουμε μια εικόνα του τι πραγματικά παράγει και διανέμει. Έτσι, η δομή της με βάση τα παραπάνω διαμορφώνεται όπως φαίνεται στο Σχήμα 2.



Σχήμα 2: Γεννήτρια Ιχνών

3.2.2 Μονάδα Εισόδου

Η μονάδα εισόδου στην πραγματικότητα είναι ένα σύνολο ανεξάρτητων υποσυστημάτων, που το καθένα διαχειρίζεται από μία πηγή. Αυτό εξασφαλίζει την επεκτασιμότητα και συντηρησιμότητα της Μονάδας Εισόδου, με την απλή προσθήκη, αφαίρεση ή τροποποίηση υποσυστημάτων. στο Σχήμα 3 φαίνεται η δομή της.



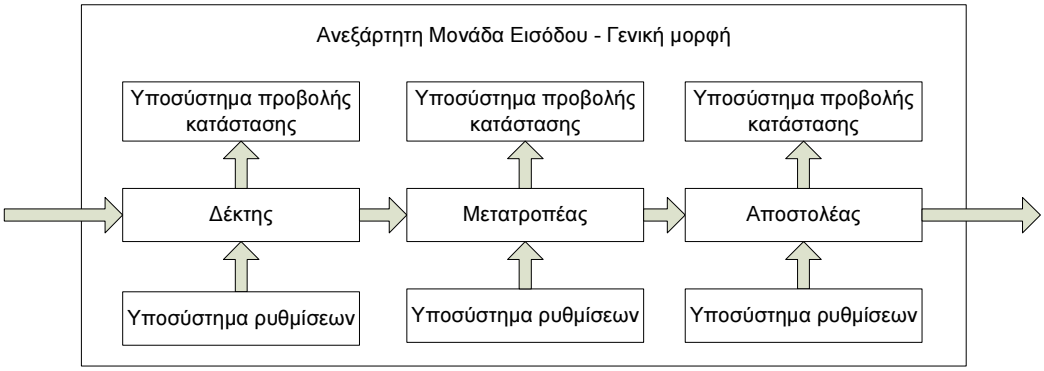
Σχήμα 3: Δομή Μονάδας Εισόδου

3.2.2.1 Ανεξάρτητη Μονάδα Εισόδου

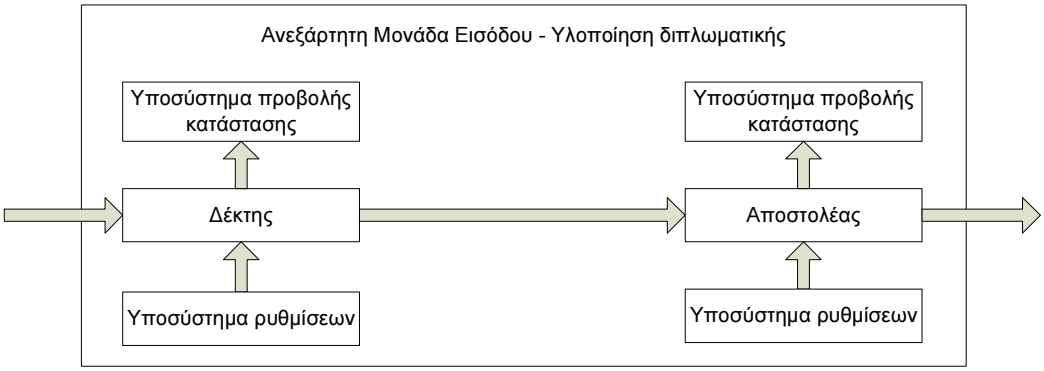
Κάθε Ανεξάρτητη Μονάδα Εισόδου μπορεί να χωριστεί με τη σειρά της σε εννέα επιμέρους υποσυστήματα. Το υποσύστημα λήψης ιχνών από την εξωτερική πηγή (Δέκτης), το υποσύστημα μετατροπής/επεξεργασίας των ιχνών (Μετατροπέας), το υποσύστημα αποστολής των ιχνών (Αποστολέας), τρία υποσυστήματα προβολής της κατάστασης του Δέκτη, του Μετατροπέα και του Αποστολέα, και τρία υποσυστήματα ρύθμισης του Δέκτη, του Μετατροπέα και του Αποστολέα. Στο Σχήμα 4 παρουσιάζεται η εσωτερική δομή μιας Ανεξάρτητης Μονάδας Εισόδου.

Στην εργασία αυτή, η Ανεξάρτητη Μονάδα Εισόδου που αναπτύχθηκε, δεν έχει Μετατροπέα (ούτε το υποσύστημα προβολής της κατάστασής του, ούτε το υποσύστημα ρύθμισής του) καθώς τα ίχνη που λαμβάνουμε από τη Γεννήτρια Ιχνών είναι ήδη στη μορφή που θέλουμε,

και δεν απαιτείται περαιτέρω επεξεργασία. Στο Σχήμα 5 φαίνεται η εσωτερική δομή της υλοποιημένης Ανεξάρτητης Μονάδας Εισόδου.



Σχήμα 4: Ανεξάρτητη Μονάδα Εισόδου - Γενική μορφή



Σχήμα 5: Ανεξάρτητη Μονάδα Εισόδου - Υλοποίηση εργασίας

Χωρίζοντας την Ανεξάρτητη Μονάδα Εισόδου με αυτό τον τρόπο, επιτυγχάνουμε ευκολία σε μελλοντική αναβάθμιση ή τροποποίηση της λειτουργίας οποιουδήποτε τμήματος, ώστε να καλύψει τυχόν μελλοντικές απαιτήσεις.

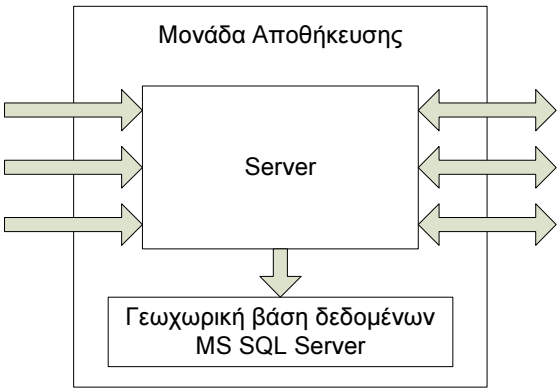
3.2.3 Μονάδα Αποθήκευσης

Η εφαρμογή όπως φαίνεται από τις απαιτήσεις τις που παρουσιάστηκαν, θα έχει αυξημένες ανάγκες αποθήκευσης / ανάκτησης δεδομένων. Θα ήταν αναμενόμενο η Μονάδα Αποθήκευσης να είναι ένα από τα πιο πολύπλοκα σημεία της εφαρμογής. Αυτό δε συμβαίνει για τους εξής λόγους:

1. Είναι δεδομένο όπως προκύπτει από τους περιορισμούς που παρουσιάστηκαν στην παράγραφο 1.2 ότι η γεωχωρική βάση δεδομένων θα είναι η Microsoft SQL Server. Οι δυνατότητές της είναι δεδομένες και δεν είναι εφικτό να διαφοροποιηθεί με κάποιο τρόπο.

2. Λαμβάνοντας υπόψη την παραπάνω παράγραφο, ο μόνος τρόπος αύξησης των δυνατοτήτων της Μονάδας Αποθήκευσης, θα ήταν η ανάπτυξη ενός service που θα μπορούσε να κατανέμει την πληροφορία σε αρκετές βάσεις δεδομένων σε διαφορετικά μηχανήματα, βάση κάποιου αλγορίθμου. Κάτι τέτοιο όμως είναι πολύ εκτενές ώστε να υλοποιηθεί στην παρούσα εργασία. Αυτό που υλοποιήθηκε, είναι το σύνολο της εφαρμογής να είναι modular, με αποτέλεσμα να μπορεί, αν απαιτηθεί, να αντικατασταθεί η Μονάδα Αποθήκευσης ή τμήμα της με κάποια υλοποίηση αυξημένων δυνατοτήτων.
3. Αν κατά τον έλεγχο αποδειχθεί ότι απαιτείται καλύτερη Μονάδα Αποθήκευσης, τότε θα μπορούσε να τροποποιηθεί ο Αποστολέας κάθε Ανεξάρτητης Μονάδας Εισόδου ώστε να μοιράζει τα ίχνη σε διαφορετικές Μονάδες Αποθήκευσης. Και αν οι υλοποιήσεις Ανεξάρτητων Μονάδων Εισόδου είναι πολλές και δύσκολο να τροποποιηθούν, τότε θα μπορούσε να προστεθεί ένα ακόμα επίπεδο αφαίρεσης που να περιβάλει τις Μονάδες Αποθήκευσης, με σκοπό να κάνει την κατανομή και ανασύνθεση των ιχνών από αυτές.

Κατά συνέπεια, η δομή που υιοθετήθηκε είναι απλοϊκή, όπως φαίνεται στο Σχήμα 6 με την προσδοκία ότι θα είναι επαρκής.

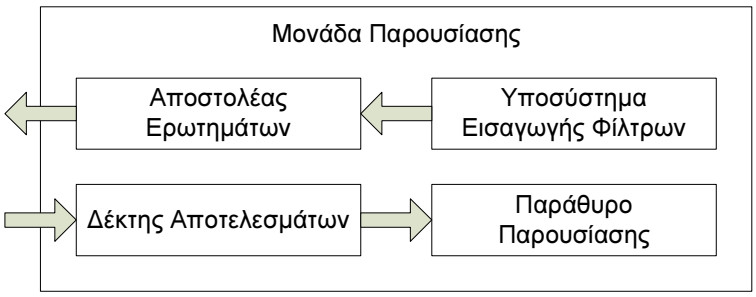


Σχήμα 6: Δομή Μονάδας Αποθήκευσης

3.2.4 Μονάδα Παρουσίασης

Αυτό το υποσύστημα είναι υπεύθυνο για την παρουσίαση των ιχνών που έχουν συλλεχθεί στον τελικό χρήστη. Οι επιχειρησιακές απαιτήσεις δεν είναι με ακρίβεια καθορισμένες. Αρκεί να έχει τη δυνατότητα παρουσίασης της παρούσας κατάστασης του εναέριου χώρου και της κατάστασης του εναέριου χώρου σε κάποια καθορισμένη στιγμή στο παρελθόν. Και στις δύο περιπτώσεις θα πρέπει να μπορούν να εισάγονται φίλτρα που θα περιορίζουν το πλήθος των ιχνών που θα προβάλλεται. Επομένως τα επιμέρους τμήματά της είναι ένα Υποσύστημα Εισαγωγής Φίλτρων, ένα υποσύστημα που θα ζητάει από τη Μονάδα Αποθήκευσης τα ίχνη

που επιθυμούμε (Αποστολέας Ερωτημάτων), ένας Δέκτης Αποτελεσμάτων που θα λαμβάνει τα ίχνη που ζητήθηκαν από τη Μονάδα Αποθήκευσης, και τέλος ένα Παράθυρο Παρουσίασης όπου θα προβάλλονται τα ίχνη που έχουν ληφθεί. Η δομή αυτή φαίνεται στο Σχήμα 7. Αξίζει να σημειωθεί εδώ ότι παρόλο που δε φαίνεται στο Σχήμα 7, ο Αποστολέας Ερωτημάτων, στην περίπτωση που θα ζητήσουμε την παρούσα κατάσταση του εναέριου χώρου, θα πρέπει χωρίς περαιτέρω επέμβαση του χρήστη να επαναλαμβάνει ανά τακτά χρονικά διαστήματα το αρχικό ερώτημα για διαφορετικό χρονικό παράθυρο κάθε φορά, ώστε να ανανεώνεται αυτόματα η εικόνα που δίνεται στον τελικό χρήστη.



Σχήμα 7: Δομή Μονάδας Παρουσίασης

3.3 Μοντέλο Οντοτήτων Συσχετίσεων

Το μοντέλο οντοτήτων - συσχετίσεων (ER model) της βάσης δεδομένων δε θα το παρουσιάσουμε καν σχηματικά, καθώς είναι όσο πιο απλό γίνεται. Αποτελείται από μία και μόνη οντότητα που χρησιμοποιείται για την αποθήκευση των ιχνών. Κάθε ίχνος αποτελείται από πληροφορίες που μπορούν να αλλάξουν ανά πάσα στιγμή και κάθε πληροφορία που έχει μικρό και πεπερασμένο πλήθος τιμών είναι πολύ μικρή σε μέγεθος ώστε να χρειάζεται να αποθηκεύονται οι πιθανές τιμές της σε διαφορετική οντότητα. Σε περίπτωση που μελλοντικά ενσωματωθούν περισσότερες λειτουργίες του PALLAS στο παρόν σύστημα, το μοντέλο οντοτήτων - συσχετίσεων θα αποκτήσει κάποια αξιοσημείωτη διάσταση. Στα πλαίσια όμως αυτής της εργασίας, δεν έχει τίποτα το αξιοσημείωτο να επιδείξει.

4

Πλατφόρμες, εργαλεία, πρότυπα

Στο προηγούμενο κεφάλαιο είδαμε την αρχιτεκτονική του συστήματος. Αναλύσαμε τα υποσυστήματά του ώστε να δούμε τι κάνει κάθε ένα από αυτά. Στο επόμενο κεφάλαιο θα δούμε πώς η παραπάνω αρχιτεκτονική υλοποιήθηκε στην πράξη.

Πριν προχωρήσουμε όμως, θα πρέπει να δούμε τις πλατφόρμες και τα προγραμματιστικά εργαλεία που επιλέχθηκαν και χρησιμοποιήθηκαν, καθώς αυτές οι επιλογές επηρεάζουν και τον τρόπο υλοποίησης που ακολουθεί. Είναι μάλιστα απαραίτητη η κατανόηση ορισμένων από τις ακόλουθες παραγράφους, προκειμένου να αντιληφθεί κανείς τους λόγους που η υλοποίηση συμμορφώνεται με την αρχιτεκτονική που έχει περιγραφεί, και τις απαιτήσεις που έχουν τεθεί.

4.1 Microsoft SQL Server 2008 R2 Express

Σύμφωνα με τις απαιτήσεις, η εφαρμογή θα πρέπει να χρησιμοποιεί ως γεωχωρική βάση δεδομένων την Microsoft SQL Server. Η έκδοση για την οποία έχει ήδη άδειες χρήσης η ΠΑ, είναι η 2008 R2. Στην εργασία αυτή χρησιμοποιήθηκε η 2008 R2 Express, η οποία υστερεί σε ορισμένα σημεία, αλλά κατά τα άλλα είναι πανομοιότυπη. Το δύο σημεία που υστερεί και ίσως μας επηρεάζει, είναι ότι έχει μέγιστο μέγεθος βάσης δεδομένων τα 10GB και μέγιστο χρησιμοποιούμενο μέγεθος μνήμης το 1GB. Κατά τα άλλα, αν η εγκατάσταση γίνει στην πλήρη έκδοση, δε χρειάζεται να αλλάξει ούτε μία γραμμή κώδικα, ούτε μία ρύθμιση. Τα

χαρακτηριστικά και τις δυνατότητες της βάσης, καθώς και το ίδιο το λογισμικό, μπορεί να τα βρει κανείς στο (Microsoft, 2012a).

4.2 Web Feature Service

Τα Web Feature Services (WFS) (OGC, 2012a) είναι ένα πρότυπο του Open Geospatial Consortium (OGC, 2012b). Είναι ένα διεθνώς αναγνωρισμένο πρότυπο που περιγράφει τη διεπαφή που θα πρέπει να υλοποιήσει ένας server προκειμένου να μπορεί κάποιος απομακρυσμένα να διαχειριστεί τη χωρική πληροφορία που περιέχει. Η εφαρμογή απαιτείται στο υποσύστημα του server να υποστηρίζει απομακρυσμένη εισαγωγή γεωχωρικών δεδομένων, και απομακρυσμένη λήψη γεωχωρικών δεδομένων, και όπως είδαμε στην παράγραφο 1.2, η χρήση αναγνωρισμένων προτύπων όπου είναι δυνατόν είναι μέρος των απαιτήσεων. Επομένως, αποφασίστηκε ο server να υλοποιεί WFS και όλα τα υποσυστήματα που επικοινωνούν με αυτόν να το κάνουν χρησιμοποιώντας WFS.

4.3 GeoServer

Στη θέση του server, προκειμένου να υλοποιηθεί το πρότυπο WFS έπρεπε να τοποθετηθεί λογισμικό που να το υποστηρίζει, και πάντα σύμφωνα με τις απαιτήσεις της εργασίας, να έχει όσο το δυνατόν χαμηλότερο κόστος. Με βάση αυτά τα κριτήρια επιλέχθηκε ο GeoServer (bowens et all, 2009), που αποτελεί πρόγραμμα του Open Source Geospatial Foundation (OSGeo). Ο GeoServer ισχυρίζεται ότι υλοποιεί πλήρως τα πρότυπα WCS 1.0, WMS 1.1.1, και WFS 1.0, ενώ ταυτόχρονα διανέμεται δωρεάν. Έχει μάλιστα χρησιμοποιηθεί σε πληθώρα εφαρμογών, πολλές από αυτές εμπορικές. Επιπλέον υποστηρίζει αποθήκευση γεωχωρικής πληροφορίας σε μια πληθώρα μορφών, μία από τις οποίες είναι γεωχωρική βάση δεδομένων σε Microsoft SQL Server. Άρα είναι ένα δοκιμασμένο, δωρεάν, συμβατό με WFS και MS SQL Server προϊόν. Σύμφωνα με τα παραπάνω είναι ότι ακριβώς απαιτείται.

4.4 GeoTools

Στο άλλο άκρο της επικοινωνίας, η χρήση εργαλείων που θα μπορούν να αξιοποιήσουν το WFS για να στείλουν ή να λάβουν δεδομένα είναι απαραίτητη. Επιπλέον, για την απεικόνιση των ιχνών στο Παράθυρο Παρουσίασης είναι επίσης απαραίτητη η χρήση εργαλείων που να μπορούν να απεικονίσουν σωστά γεωαναφερμένη πληροφορία (τα ίχνη) αλλά και ένα στοιχειώδες υπόβαθρο. Συνδυάζοντας και το χαμηλό κόστος, η λύση που επιλέχθηκε είναι τα GeoTools (GeoTools, 2012a), ένα ακόμα πρόγραμμα του OSGeo, στο οποίο μάλιστα βασίστηκε και η ανάπτυξη του GeoServer.

4.5 NetBeans

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε η γλώσσα προγραμματισμού Java και συγκεκριμένα το Java Development Kit (JDK) που αποτελεί μέρος της Java SE 7u9 (Oracle, 2012a). Για την εκμάθηση της γλώσσας και βοήθεια σχετικά με τη χρήση της χρησιμοποιήθηκε το (Oracle, 2012, June). Ως περιβάλλον ανάπτυξης επιλέχθηκε το NetBeans Integrated Development Environment (Oracle, 2012b). Όλα τα παραπάνω είναι διαθέσιμα δωρεάν, αναγνωρισμένα, δοκιμασμένα και αποτελεσματικά.

Όπως είδαμε από τη δομή της εφαρμογής, σε αρκετά σημεία χρειάζεται να προβάλλονται στο χρήστη κάποιες πληροφορίες, με αποκορύφωμα την ίδια την παρουσίαση των ιχνών. Επομένως, η χρήση ενός παραθυρικού περιβάλλοντος ήταν επιβεβλημένη. Είναι συνηθισμένο φαινόμενο, στις εφαρμογές που χρησιμοποιούν παραθυρικό περιβάλλον, οι προγραμματιστές να αφιερώνουν ένα πολύ μεγάλο ποσοστό του χρόνου τους στην ανάπτυξη της διεπαφής με το χρήστη, και συγκεκριμένα στη σωστή διαχείριση (δημιουργία, άνοιγμα, κλείσιμο και σύνδεση) των στοιχείων που εμφανίζονται στην οθόνη (παράθυρα, φόρμες, μενού, γραμμές κατάστασης κ.α.). Για την αποφυγή σπατάλης χρόνου σε τέτοιες βασικές και επαναλαμβανόμενες εργασίες, δημιουργήθηκε από την Oracle μια πλατφόρμα ανάπτυξης εφαρμογών με το όνομα NetBeans Platform (Oracle, 2012c). Η πλατφόρμα αυτή θεωρήθηκε ιδανική για τη συγκεκριμένη εφαρμογή, γιατί αναγκαστικά οι εφαρμογές που αναπτύσσονται με αυτή έχουν modular μορφή. Δηλαδή, προσφέρει όλα τα απαραίτητα εργαλεία ώστε να αναπτύξεις μια εφαρμογή που είναι χωρισμένη σε modules, τα οποία έχουν σχέσεις εξάρτησης μεταξύ τους. Κάθε module έχει τη δυνατότητα να εγκαθίσταται στην εφαρμογή εκ των υστέρων, ενεργοποιείται μόνο αν στην εφαρμογή είναι εγκατεστημένα όλα τα modules από τα οποία εξαρτάται, με την εγκατάστασή του επεκτείνεται το μενού και η γραμμή εργαλείων με τις δυνατότητες που αυτό προσφέρει, και έχει ένα πλήθος άλλων χαρακτηριστικών. Αυτή η δομή κάνει την επέκταση, συντήρηση και τροποποίηση της εφαρμογής εύκολη, χωρίς να αλλάξει ούτε μία γραμμή από τον κώδικα που έχει ήδη γραφτεί.

Κάθε module έχει εσωτερικά τη δομή μιας μικρής εφαρμογής. Επικοινωνεί με τα υπόλοιπα modules μέσω του Lookup. Με απλά λόγια το Lookup είναι ένας μια δομή που αποθηκεύει αναφορές σε αντικείμενα. Ένα module μπορεί να προσθέσει αντικείμενα στο Lookup ή να εκτελέσει μια αναζήτηση για αντικείμενα μιας συγκεκριμένης κλάσης, ή που υλοποιούν ένα συγκεκριμένο interface. Μπορεί επίσης να ζητήσει να ενημερώνεται αυτόματα από το Lookup όταν υπάρξει μια αλλαγή στα περιεχόμενά του που επιφέρει αλλαγές σε μια συγκεκριμένη αναζήτηση. Ακούγεται λίγο πολύπλοκο, αλλά στην πραγματικότητα είναι μια απλοποιημένη υλοποίηση μιας event-driven εφαρμογής με επιπλέον δυνατότητες. Στο

επόμενο κεφάλαιο όπου θα δούμε πώς η εφαρμογή χρησιμοποιεί τη modular δομή και το Lookup, θα γίνει πιο κατανοητή η λειτουργία του.

4.6 Δεδομένα και βιβλιοθήκες

Για τη σύνδεση του GeoServer με τον Microsoft SQL Server χρησιμοποιήθηκε το Microsoft JDBC Driver 4.0 for SQL Server που είναι διαθέσιμο στο (Microsoft, 2012b). Για το υπόβαθρο στο Παράθυρο Παρουσίασης χρησιμοποιήθηκαν δεδομένα από το (DIVA-GIS). Τέλος, για την επιλογή της ημερομηνίας στο υποσύστημα εισαγωγής φίλτρων χρησιμοποιήθηκε το JCalendar από (Kai Tödter, 2012).

5

Σχεδίαση Συστήματος

Σε αυτό το σημείο έχουμε ήδη δει το σκοπό της εργασίας, τους βασικούς περιορισμούς, την αρχιτεκτονική του συστήματος όπως αυτή πηγάζει από τα παραπάνω, και τέλος είδαμε τα εργαλεία που επιλέχθηκαν για την υλοποίηση της εφαρμογής. Σε αυτό το κεφάλαιο θα δούμε σε γενικά πλαίσια, πως η αρχιτεκτονική του συστήματος υλοποιήθηκε στην εφαρμογή με τη χρήση των εργαλείων που αναφέρθηκαν.

5.1 Αρχιτεκτονική

Η γενική αρχιτεκτονική της υλοποίησης είναι παραπλήσια με την αρχιτεκτονική που περιγράψαμε στην ανάλυση των απαιτήσεων του κεφαλαίου 3. Στις περισσότερες περιπτώσεις, κάθε υποσύστημα που περιγράψαμε στην ανάλυση, αντιστοιχεί σε ένα module στην εφαρμογή. Υπάρχουν κάποια επιπλέον modules, και κάποια υποσυστήματα (όπως η βάση δεδομένων) που υλοποιούνται από έτοιμο λογισμικό.

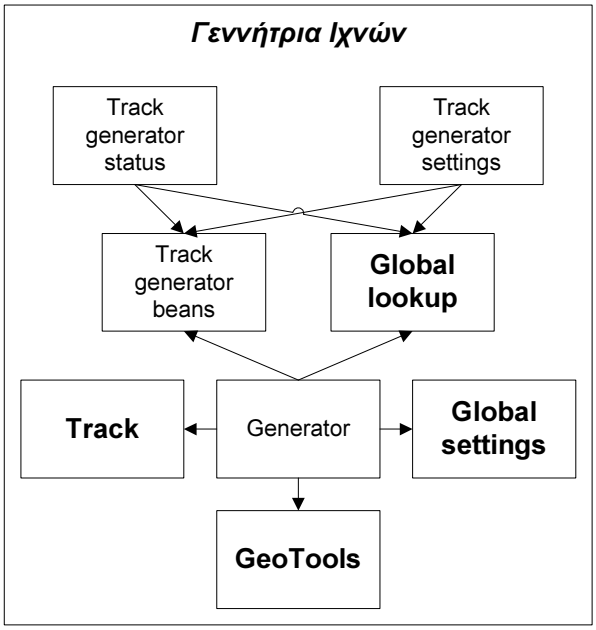
Στις επόμενες παραγράφους θα δούμε ποια modules απαρτίζουν κάθε ένα από τα τέσσερα υποσυστήματα της εφαρμογής, και στη συνέχεια θα δούμε ποιες κλάσεις απαρτίζουν κάθε module.

5.2 Υποσυστήματα

Εκτός από τη Μονάδα Αποθήκευσης, τα υπόλοιπα υποσυστήματα απαρτίζονται από modules. Ορισμένα από αυτά είναι κοινά σε όλα τα υποσυστήματα. Εδώ θα δούμε ποια modules απαρτίζουν τα τρία υποσυστήματα και ποιες οι ρυθμίσεις για τη Μονάδα Αποθήκευσης.

5.2.1 Γεννήτρια Ιχνών

Τα modules που συνθέτουν τη Γεννήτρια Ιχνών φαίνονται στο Σχήμα 8. Τα βέλη δείχνουν σχέση εξάρτησης.



Σχήμα 8: Modules της Γεννήτριας Ιχνών

Τα modules με bold γράμματα χρησιμοποιούνται και σε άλλα υποσυστήματα. Το module Generator αντιστοιχεί στη Γεννήτρια στο Σχήμα 2. Το module Track generator status αντιστοιχεί στο Υποσύστημα προβολής κατάστασης στο Σχήμα 2, ενώ το Track generator settings στο Υποσύστημα ρυθμίσεων.

Το module Track περιέχει την μορφή της εσωτερικής απεικόνισης ενός ίχνους, το GeoTools περιέχει τις εξωτερικές βιβλιοθήκες του GeoTools, το GlobalSettings περιέχει καθολικές ρυθμίσεις, και το Global lookup περιέχει ένα Lookup που χρησιμοποιείται από διάφορα modules της εφαρμογής.

Η Γεννήτρια Ιχνών είναι ένα καλό παράδειγμα του πως η modular δομή διευκολύνει μελλοντικές αναβαθμίσεις και τροποποιήσεις. Το module Track generator settings παρέχει ρυθμίσεις για το Generator, χωρίς να εξαρτάται το ένα από το άλλο. Τα κοινά στοιχεία τους

είναι δύο άλλα modules. Το Track generator beans δεν είναι τίποτα άλλο από δύο κλάσεις που περιγράφουν δύο δομές: μία για τις ρυθμίσεις και μία για την κατάσταση. Όταν ο χρήστης επιλέξει να ενεργοποιήσει τη γεννήτρια, το Track generator settings δημιουργεί ένα instance της μίας κλάσης και τη βάζει στο Global Lookup. Το module Generator ενημερώνεται από το Lookup ότι τοποθετήθηκε σε αυτό μια κλάση που το ενδιαφέρει, διαβάζει από το Lookup τις νέες ρυθμίσεις, και τις εφαρμόζει. Κάτι αντίστοιχο συμβαίνει με την κατάσταση.

Αν τώρα κάποιος επιθυμεί μελλοντικά να βελτιώσει ένα από τα τρία module, δε χρειάζεται να αλλάξει καθόλου τα άλλα δύο. Το Global Lookup και το Track generator beans λειτουργούν ως ενδιάμεσος κρίκος που τα αποσυνδέει. Παρόμοια λειτουργικότητα προσδίδει και ένα interface της Java. Μόνο που με το NetBeans Platform, η προσθήκη, αφαίρεση και αναβάθμιση κάποιου module μπορεί να γίνει και αφού η εφαρμογή έχει μοιραστεί και χρησιμοποιείται, ενώ τρέχει, και με αυτόματο έλεγχο για updates, ενώ το interface σε διευκολύνει μόνο πριν η εφαρμογή δοθεί στον τελικό χρήστη.

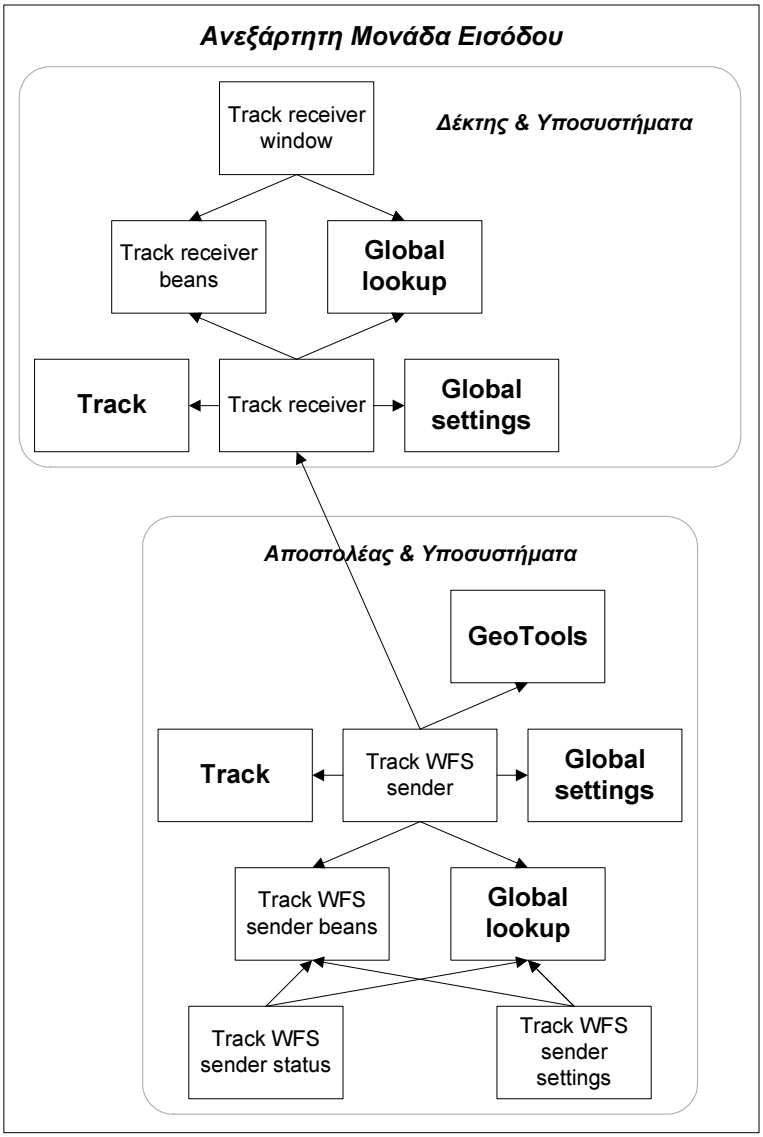
Ολόκληρη η Γεννήτρια Ιχνών είναι αποσυνδεμένη από την υπόλοιπη εφαρμογή και μπορεί να εκτελείται σε διαφορετικό μηχάνημα.

5.2.2 Μονάδα Εισόδου

Η Μονάδα Εισόδου είναι μια οντότητα που προγραμματιστικά δεν υφίσταται. Αποτελεί εννοιολογικά το σύνολο των Ανεξάρτητων Μονάδων Εισόδου που υπάρχουν. Επομένως, αρκεί να αναλύσουμε μία Ανεξάρτητη Μονάδα Εισόδου. Συγκεκριμένα, η μορφή που έχει η υλοποίησή μας φαίνεται στο Σχήμα 9. Τα βέλη δείχνουν σχέση εξάρτησης.

Το module Track receiver είναι ο Δέκτης στο Σχήμα 5, το Track receiver window είναι το ο συνδυασμός του Υποσυστήματος ρυθμίσεων και του Υποσυστήματος προβολής κατάστασης στο Σχήμα 5, το Track WFS sender είναι ο Αποστολέας, το Track WFS sender settings είναι το Υποσύστημα ρυθμίσεων, και τέλος το Track WFS status είναι το Υποσύστημα προβολής κατάστασης. Τα τέσσερα module με bold είναι τα ίδια με της προηγούμενης παραγράφου.

Το μοντέλο που ακολουθήσαμε στη Γεννήτρια Ιχνών για την αποσύνδεση των modules ακολουθείται και εδώ. Η διαφοροποίηση είναι ότι λόγω των λίγων στοιχείων που χρειάζονται για να ρυθμιστεί ο Track receiver και για να προβάλει την κατάστασή του, ένα παράθυρο είναι αρκετό. Φαίνεται μάλιστα σε αυτή την υλοποίηση πόσο εύκολα μπορεί να προσαρμοστεί η δομή της εφαρμογής σε διαφορετικές απαιτήσεις. Τα modules Track receiver beans και Track WFS sender beans έχουν τον ίδιο ρόλο με το Track generator beans της προηγούμενης παραγράφου.



Σχήμα 9: Modules της Ανεξάρτητης Μονάδας Εισόδου

Το μοντέλο αυτό, παρότι αποτελεί πλεονέκτημα στις παραπάνω περιπτώσεις, δε χρησιμοποιήθηκε για την αποσύνδεση του Track receiver από τον Track WFS sender. Ο λόγος είναι ότι το Global lookup δεν είναι τόσο γρήγορο, ούτε έχει σχεδιαστεί με τέτοιο τρόπο ώστε να λειτουργεί σαν buffer. Τα δύο module συνδέονται μόνο και μόνο γιατί πρέπει να ανταλλάσουν πληροφορίες (ίχνη). Δε χρησιμοποιούν κατά τα άλλα λειτουργικότητα από το άλλο module. Επομένως, αν συνδέονταν μέσω του Lookup, θα έπρεπε το Lookup να μεταφέρει ουσιαστικά τα δεδομένα από το ένα module στο άλλο, δηλαδή να λειτουργεί ως buffer. Και όπως είπαμε αυτό δε γίνεται. Υπάρχουν και άλλες μέθοδοι που θα μπορούσαν να αποσυνδέσουν τα δύο modules, όπως η δημιουργία ενός ενδιάμεσου module που θα υλοποιεί το buffer ή να επικοινωνούν μέσω δικτύου (οπότε θα μπορούσαν να μοιραστούν σε

διαφορετικά μηχανήματα), αλλά θεωρήθηκε ότι είναι προτιμότερο να μην προστεθεί επιπλέον πολυπλοκότητα που θα έχει επιπτώσεις στις επιδόσεις.

5.2.3 Μονάδα Αποθήκευσης

Για τη Μονάδα Αποθήκευσης δεν γράφτηκε καθόλου κώδικας. Δημιουργήθηκε η βάση δεδομένων όπως περιγράφεται στην παράγραφο 5.4, και ρυθμίστηκε ο GeoServer έτσι ώστε να μπορεί κανείς με WFS να προσθέσει ή να ανακτήσει δεδομένα από αυτή. Στην πορεία βέβαια αποκαλύφθηκε ότι ο GeoServer, προκειμένου να συνδεθεί σωστά με την βάση, έπρεπε να τροποποιηθεί το σχήμα της βάσης (τα πεδία του μοναδικού πίνακα). Έτσι η βάση ξανασχεδιάστηκε. Λεπτομέρειες υπάρχουν στην παράγραφο 5.4.

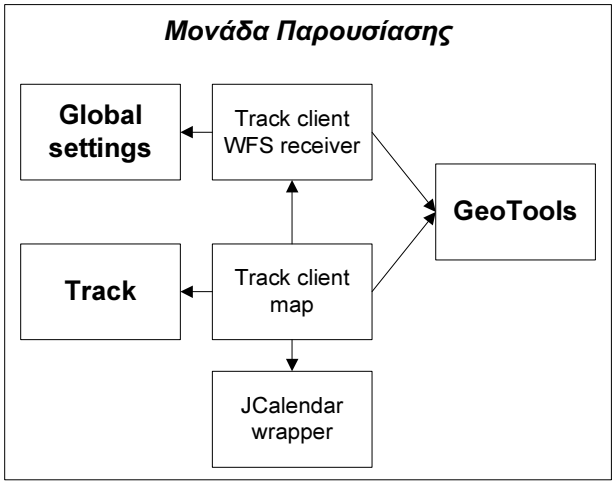
Τα ίχνη έχουν το καθένα από τη δημιουργία τους ένα πεδίο που δείχνει τη χρονική στιγμή δημιουργίας τους, με ακρίβεια millisecond. Έτσι, αν κάποιος ζητήσει τα ίχνη μιας δεδομένης στιγμής, το πιθανότερο είναι ότι δε θα πάρει κανένα αποτέλεσμα, καθώς είναι απίθανο εκείνο το συγκεκριμένο ms να είχε δημιουργηθεί κάποιο ίχνος. Αντί για αυτό, η αναζήτηση γίνεται πάντα σε ένα παράθυρο χρόνου. Το παράθυρο αυτό μάλιστα πρέπει να έχει διάρκεια αρκετά δευτερόλεπτα. Ο λόγος είναι ότι η κεραία που ανιχνεύει ένα αεροσκάφος και παράγει ένα ίχνος, περιστρέφεται αργά. Εκτελεί στην καλύτερη περίπτωση μία περιστροφή ανά 5 sec. Άρα αν θέλουμε να μας επιστραφεί το τελευταίο ίχνος για κάθε αεροσκάφος, θα πρέπει να αναζητήσουμε ίχνη τουλάχιστον 5 sec παλιά. Επιπλέον, αφού υπάρχει το ενδεχόμενο να μην ανιχνευτεί το αεροσκάφος σε κάποια περιστροφή (και να μην παραχθεί ίχνος), αλλά και επειδή ορισμένες κεραίες εκτελούν μια περιστροφή ανά 20 sec, θα πρέπει να αναζητήσουμε ίχνη τουλάχιστον 40 sec παλιά. Μέσα σε ένα τόσο μεγάλο χρονικό παράθυρο όμως, για πολλά αεροσκάφη θα έχουν καταχωρηθεί περισσότερα από ένα ίχνη. Επομένως είναι απαραίτητο κάθε φορά να γίνεται ένα φιλτράρισμα με το οποίο μέσα από το χρονικό παράθυρο που θα ζητήσουμε, θα μας επιστρέφεται μόνο το πιο πρόσφατο ίχνος για κάθε αεροσκάφος. Αυτό μπορεί να γίνει με ένα SQL ερώτημα, μόνο που ο GeoServer δεν υποστηρίζει Views του Microsoft SQL Server (έτοιμα ερωτήματα). Η λύση είναι το ερώτημα να τοποθετηθεί στον ίδιο το GeoServer, ο οποίος το προωθεί στη βάση δεδομένων. Το ίδιο το ερώτημα θα το δούμε επίσης στην παράγραφο 5.4, μαζί με τη βάση δεδομένων.

Ένα ακόμα ενδιαφέρον σημείο είναι ότι ο GeoServer υποστηρίζει ερωτήματα ανάκτησης δεδομένων σύμφωνα με το πρωτόκολλο WFS 1.1.0, αλλά για ερωτήματα εισαγωγής δεδομένων, υποστηρίζει μόνο το WFS 1.0.0. Στην περίπτωσή μας δε μας επηρεάζει με κάποιο τρόπο.

Οι λεπτομέρειες για τη ρύθμιση του GeoServer παρουσιάζονται στην παράγραφο 6.2.

5.2.4 Μονάδα Παρουσίασης

Τα modules που αποτελούν τη Μονάδα Παρουσίασης φαίνονται στο Σχήμα 10. Τα βέλη δείχνουν σχέση εξάρτησης.



Σχήμα 10: Modules της Μονάδας Παρουσίασης

Το module Track client WFS receiver είναι η συγχώνευση του Αποστολέα Ερωτημάτων και του Δέκτη Αποτελεσμάτων στο Σχήμα 7. Το Track client map είναι η συγχώνευση του Υποσυστήματος Εισαγωγής Φίλτρων και του Παράθυρου Παρουσίασης. Το JCalendar wrapper είναι ένα module που μοναδικός του σκοπός είναι να περιλαμβάνει την εξωτερική βιβλιοθήκη JCalendar η οποία χρησιμοποιείται κατά την εισαγωγή των φίλτρων για την επιλογή της ημερομηνίας. Τα υπόλοιπα τρία modules είναι τα ίδια που συναντήσαμε παραπάνω.

Στη Μονάδα Παρουσίασης δε δόθηκε ιδιαίτερη έμφαση, και οι λόγοι θα αναλυθούν στην επόμενη παράγραφο και στο επόμενο κεφάλαιο όπου και θα δούμε αναλυτικότερα την υλοποίηση του κάθε module. Παρόλα αυτά, είναι από τα σημεία της εφαρμογής με τον περισσότερο κώδικα. Τα module έχουνε αρκετές κλάσεις το καθένα.

5.3 Modules

Όπως έχουμε ήδη αναφέρει, κάθε module μοιάζει με μια μικρή εφαρμογή, ή βιβλιοθήκη. Υπάρχουν modules που εμφανίζουν κάτι στην οθόνη, και άλλα που προσφέρουν λειτουργικότητα. Το NetBeans Platform φροντίζει ώστε να γίνεται αυτόματη φόρτωση και εκκίνηση των modules που εμφανίζουν κάποιο παράθυρο. Για τα υπόλοιπα, κατά την εγκατάσταση του module στην εφαρμογή (που γίνεται αυτόματα με κάθε εκκίνησή της) υπάρχει μια κλάση (installer) που υλοποιεί ένα interface του οποίου οι μέθοδοι καλούνται από την πλατφόρμα κατά την εκκίνηση και το κλείσιμο του module, οπότε εκεί έχουμε

τοποθετήσει τον ανάλογο κώδικα που θα κάνει διαθέσιμη τη λειτουργικότητα του module. Το αναφέρουμε εδώ καθώς αποτελεί τυπικό σενάριο και θα το συναντήσουμε στις επόμενες παραγράφους, όπου βλέπουμε πιο αναλυτικά το κάθε module.

Θα ξεκινήσουμε την περιγραφή των modules από αυτά που χρησιμοποιούνται σε διάφορα σημεία της εφαρμογής, αυτά που στα προηγούμενα σχήματα απεικονίζαμε με bold γράμματα.

5.3.1 Track

Αυτό το module δεν εμφανίζει κάποιο παράθυρο, ούτε προσφέρει κάποια λειτουργικότητα. Περιέχει τη δομή δεδομένων που απεικονίζει ένα ίχνος εσωτερικά στην εφαρμογή, και αποτελεί ξεχωριστό module μόνο και μόνο γιατί εννοιολογικά δεν αποτελεί μέρος κανενός υποσυστήματος, αλλά πρέπει να είναι διαθέσιμο σε πολλά από αυτά. Οι κλάσεις που περιέχει είναι οι ακόλουθες δύο, όπως απεικονίζονται στο Σχήμα 11.

Σχήμα 11: Κλάσεις του Track module

5.3.1.1 TrackDataSource.java

Κάθε ίχνος συνοδεύεται από πληροφορίες που αφορούν το κύριο radar που το ανίχνευσε. Αυτή η κλάση δεν είναι τίποτα άλλο παρά η απεικόνιση των στοιχείων του radar που αποτελεί την πηγή του ίχνους. Χρησιμοποιείται από την επόμενη κλάση.

5.3.1.2 Track.java

Αυτή η κλάση περιέχει τα στοιχεία ενός ίχνους. Όπως έχουμε αναφέρει, η εσωτερική απεικόνιση θα πρέπει να μπορεί να καλύψει όποια πιθανή πηγή συνδεθεί στο σύστημα μελλοντικά. Αυτό φυσικά κάποια στιγμή θα είναι αδύνατο, αλλά προκειμένου να πετύχουμε αυτό το στόχο τουλάχιστον για τη διάρκεια ζωής της εφαρμογής, μελετήθηκαν τρία διαφορετικά πρωτόκολλα, και η απεικόνιση δημιουργήθηκε έτσι ώστε να υπερκαλύπτει και τα τρία. Το πρώτο πρωτόκολλο ήταν το AIRCAT 500, το δεύτερο ήταν το πρωτόκολλο του PALLAS και το τρίτο ήταν το Astrerix (EUROCONTROL, 2012a) από το

European Organization for the Safety of Air Navigation. Το Astrerix μεταδίδει με διαφορά περισσότερες πληροφορίες από τα άλλα δύο και η εσωτερική απεικόνιση βασίστηκε σε αυτό, φροντίζοντας να το υπερκαλύπτει. Οι λεπτομέρειες του πρωτοκόλλου είναι διαθέσιμες στο (EUROCONTROL, 2012b). Συνοπτικά, ένα ίχνος περιέχει πληροφορίες για τη θέση (3D), την ταχύτητα, τη διεύθυνση, την πηγή, το ID του αεροσκάφους, αν πρόκειται για πραγματικό ή εξομοίωση, τον τύπο/τύπους της κεραίας που παρήγαγε το ίχνος, την απάντηση του αεροσκάφους σε Mode 2 IFF interrogation και την απάντηση σε Mode 3A interrogation, αν είναι διαθέσιμες. Λεπτομέρειες για τη σημασία του κάθε πεδίου υπάρχουν στη βιβλιογραφία του ASTERIX.

Οι πληροφορίες θέσης δεν είναι γεωαναφερμένες. Πρόκειται για μια δομή που χρησιμοποιείται πολύ, ειδικά σε buffers, και με την προσθήκη πληροφοριών γεωαναφοράς θα αύξανε πολύ το μέγεθός της, με αρνητικές επιπτώσεις στην απόδοση της εφαρμογής. Θεωρήθηκε καλύτερη λύση να υπονοείται σε όλη την έκταση της εφαρμογής, ότι το σύστημα αναφοράς που χρησιμοποιείται είναι το WGS-84 (ESPG 4326). Προτιμήθηκε αυτό το σύστημα αναφοράς παρόλο που πρόκειται για μια εφαρμογή για τον ελληνικό εναέριο χώρο, έτσι ώστε να υπάρχει η μέγιστη δυνατή συμβατότητα με οποιαδήποτε βιβλιοθήκη ή άλλη επέκταση που μπορεί να προστεθεί μελλοντικά στην εφαρμογή. Άλλωστε, αν είναι επιθυμητή η απεικόνιση σε άλλο σύστημα αναφοράς, αρκεί τα υποσυστήματα απεικόνισης να κάνουν μια απλή μετατροπή. Τα GeoTools δίνουν αυτή τη δυνατότητα χωρίς καμία προσπάθεια.

Τα πεδία που μας ενδιαφέρουν είναι το ID που θα είναι μοναδικό για κάθε αεροσκάφος αν η πηγή είναι πραγματική (στην υλοποίησή μας θα είναι τυχαίο), το longitude και latitude που όπως είπαμε θα είναι σε μοίρες στο WGS_84, το altitude που θα είναι σε μέτρα, το speed που θα είναι σε m/s, το heading που θα είναι σε μοίρες (0^o - 360^o), και τέλος το timestamp που θα είναι milliseconds από το Unix epoch.

5.3.2 GeoTools

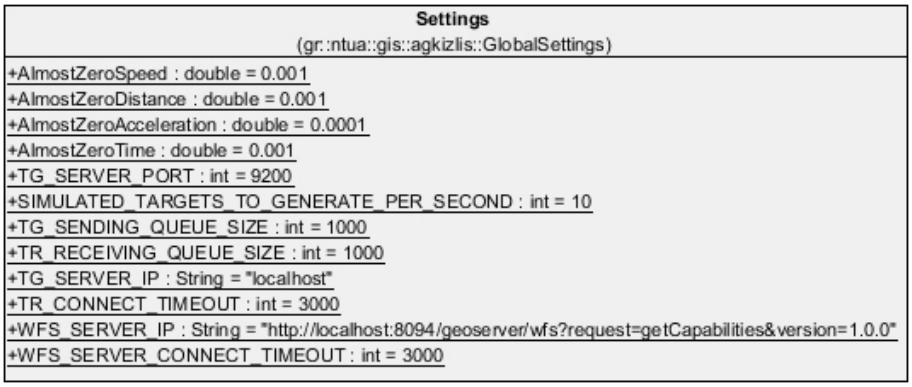
Αυτό το module δεν περιέχει καμία κλάση. Μια από τις ιδιαιτερότητες του NetBeans Platform είναι ότι αν θες μια βιβλιοθήκη να είναι διαθέσιμη σε διάφορα modules, τότε πρέπει να φτιάξεις ένα ειδικό module που θα την περιέχει (library wrapper module), και στην συνέχεια όποιο module χρειάζεται αυτή τη βιβλιοθήκη, δηλώνει εξάρτηση από το wrapper module.

Προκειμένου λοιπόν να μπορούμε να χρησιμοποιήσουμε τις βιβλιοθήκες των GeoTools, φτιάξαμε ένα library wrapper module για αυτές, το GeoTools module. Συνολικά πρόκειται για 249 αρχεία jar (βιβλιοθήκες java), με 8746 κλάσεις. Τα GeoTools μας δίνουν τη δυνατότητα να υπολογίσουμε αποστάσεις στο χώρο λαμβάνοντας υπόψη την καμπυλότητα της γης, να χειριστούμε γεωαναφερμένα δεδομένα (στη μνήμη, σε κάποιο αρχείο), να

υλοποιήσουμε WFS, WMS και WCS, να προβάλουμε γεωχωρικά δεδομένα σε κάποιο παράθυρο, αλλά και πολλές ακόμα σχετικές δυνατότητες.

5.3.3 Global Settings

Το module αυτό περιέχει μόνο μία κλάση, η οποία περιέχει σταθερές που χρησιμοποιούν κάποια άλλα modules. Έχουν συγκεντρωθεί σε ένα σημείο κάποιες τιμές που θεωρήθηκε ότι θα μπορούσαν να είναι παράμετροι. Σαν αναβάθμιση θα μπορούσε μελλοντικά αυτό το module να δίνει τη δυνατότητα αλλαγής αυτών των παραμέτρων και αποθήκευσής τους σε κάποιο αρχείο. Σαν πρώτη έκδοση, η συγκέντρωσή τους σε ένα σημείο κρίθηκε αρκετή. Στο Σχήμα 12 φαίνονται αυτές οι τιμές.

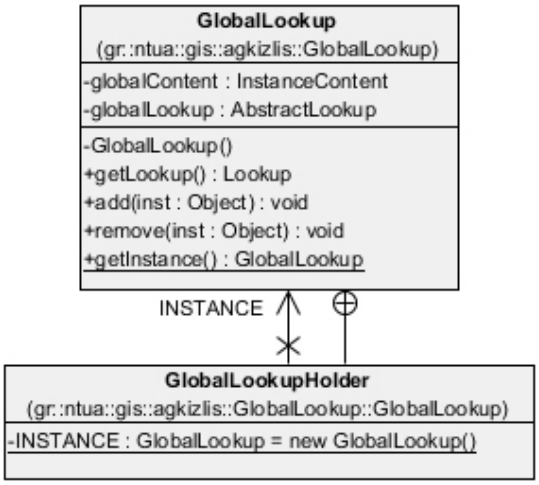


Σχήμα 12: Κλάση Settings του module Global Settings

Τα πρώτα τέσσερα πεδία καθορίζουν τις τιμές για την ταχύτητα, την απόσταση, την επιτάχυνση και τον χρόνο, κάτω από τις οποίες μια μεταβλητή θα θεωρείται μηδενική. Αυτό συμβαίνει γιατί οι πράξεις με τύπο δεδομένων double δεν είναι ποτέ ακριβείς. Τις υπόλοιπες τιμές θα τις δούμε στα module που τις χρησιμοποιούν.

5.3.4 Global Lookup

Το NetBeans Platform έχει ένα lookup προσβάσιμο από κάθε module, το οποίο έχει ένα μειονέκτημα. Τα περιεχόμενά του αλλάζουν ανάλογα με το παράθυρο που είναι ενεργό κάθε φορά. Εμείς θέλουμε ένα lookup που θα είναι διαθέσιμο από τα modules μας συνέχεια, ανεξάρτητα από το παράθυρο που είναι ενεργό. Έτσι

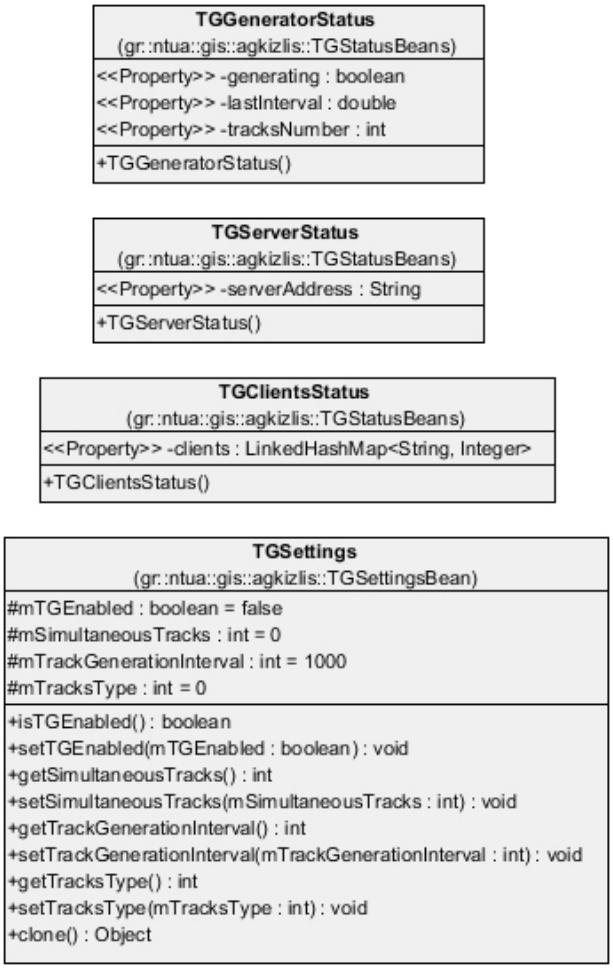


Σχήμα 13: Κλάσεις του Global lookup module

υλοποιήθηκε ένα lookup γι' αυτό ακριβώς το σκοπό, το οποίο είναι ένα singleton class, δηλαδή υπάρχει το πολύ ένα instance κάθε φορά, το οποίο δημιουργείται την πρώτη φορά που το χρειάζεσαι. Δεν έχει κάποια άλλη ιδιαιτερότητα.

5.3.5 Track generator beans

Σε αυτό το module υπάρχουν τέσσερις κλάσεις, όπως παρουσιάζονται στο Σχήμα 14. Δε θα τις παρουσιάσουμε αναλυτικά καθώς δεν έχουν κάποια λειτουργικότητα παρά να αποθηκεύουν τιμές. Όπως έχουμε ήδη αναφέρει, είναι ο συνδετικός κρίκος ανάμεσα στα module Generator, Track generator status και Track generator settings. Όταν μέσω του παραθύρου που εμφανίζει το Track generator settings module αλλάξουμε κάποια ρύθμιση για το module Generator, τότε απλά αφαιρούμε από το Global lookup το instance της κλάσης TGSettings που έχουμε βάλει εκεί (αν υπάρχει) και προσθέτουμε ένα νέο instance με τις νέες τιμές. Το Generator module παρακολουθεί τις αλλαγές στο Global lookup για την κλάση TGSettings, κι έτσι όταν βλέπει τις νέες τιμές, προσαρμόζεται ανάλογα.. Μπορεί να ρυθμιστεί ως προς το αν θα παράγει ίχνη ή όχι, πόσα αεροσκάφη θα εξομοιώνει ταυτόχρονα, κάθε πότε θα παράγει ίχνη, και τι τύπου ίχνη θα παράγει. Οι άλλες τρεις κλάσεις τοποθετούνται στο Global lookup από το Generator module, και το Track generator status module, με τη διαδικασία που προαναφέραμε, παίρνει τις νέες τιμές της κατάστασης του Generator και τις παρουσιάζει σε ένα παράθυρο. Είναι τρεις κλάσεις γιατί στην πραγματικότητα υπάρχουν τρεις λειτουργικότητες στο Generator module που «ανεβάζουν» την κατάστασή τους στο Global lookup: ο πυρήνας της γεννήτριας, ο εξυπηρετητής (που δέχεται αιτήματα δεκτών να συνδεθούν και να λαμβάνουν ίχνη), και οι

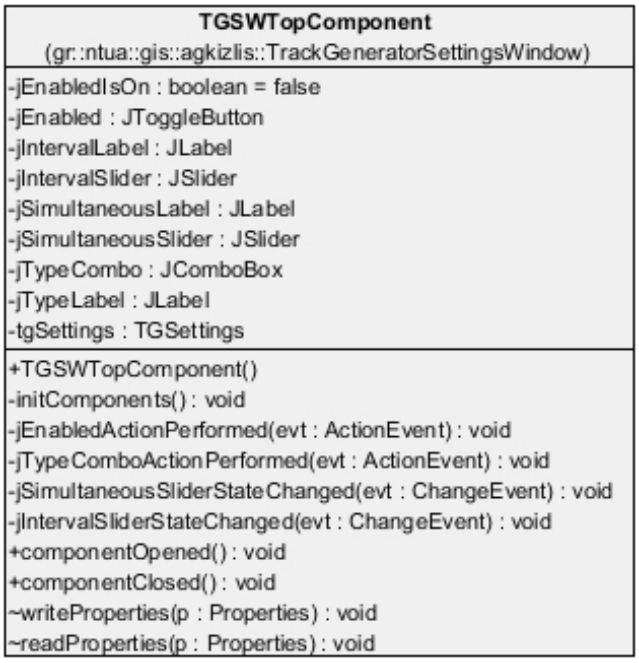


Σχήμα 14: Κλάσεις του Track generator beans

μεταδότες των ιχνών (ένας για κάθε συνδεδεμένο δέκτη). Θα δούμε περισσότερες λεπτομέρειες στην παράγραφο όπου περιγράφεται το Generator module.

5.3.6 Track generator settings

Η μοναδική κλάση του Track generator settings είναι αυτή που υλοποιεί το παράθυρο όπου εισάγονται οι ρυθμίσεις (Σχήμα 15). Υπάρχει ένα ρυθμιστικό για κάθε παράμετρο με την οποία μπορεί να ρυθμιστεί το Generator module. Οι τέσσερις διαδικασίες που δέχονται ένα event ως παράμετρο αντιστοιχούν στα τέσσερα ρυθμιστικά. Κάθε μία από αυτές, όταν κάποια ρύθμιση αλλάξει, αφαιρούν από το Global lookup το instance του TGSettings που ήδη υπάρχει εκεί (αν υπάρχει), αλλάζουν την αντίστοιχη τιμή της εσωτερικής μεταβλητής tgSettings και «ανεβάζουν» το tgSettings στο Global lookup. Στο κεφάλαιο 7 υπάρχουν και screenshots από την εφαρμογή όπου φαίνεται το interface που δημιουργεί αυτή η κλάση.



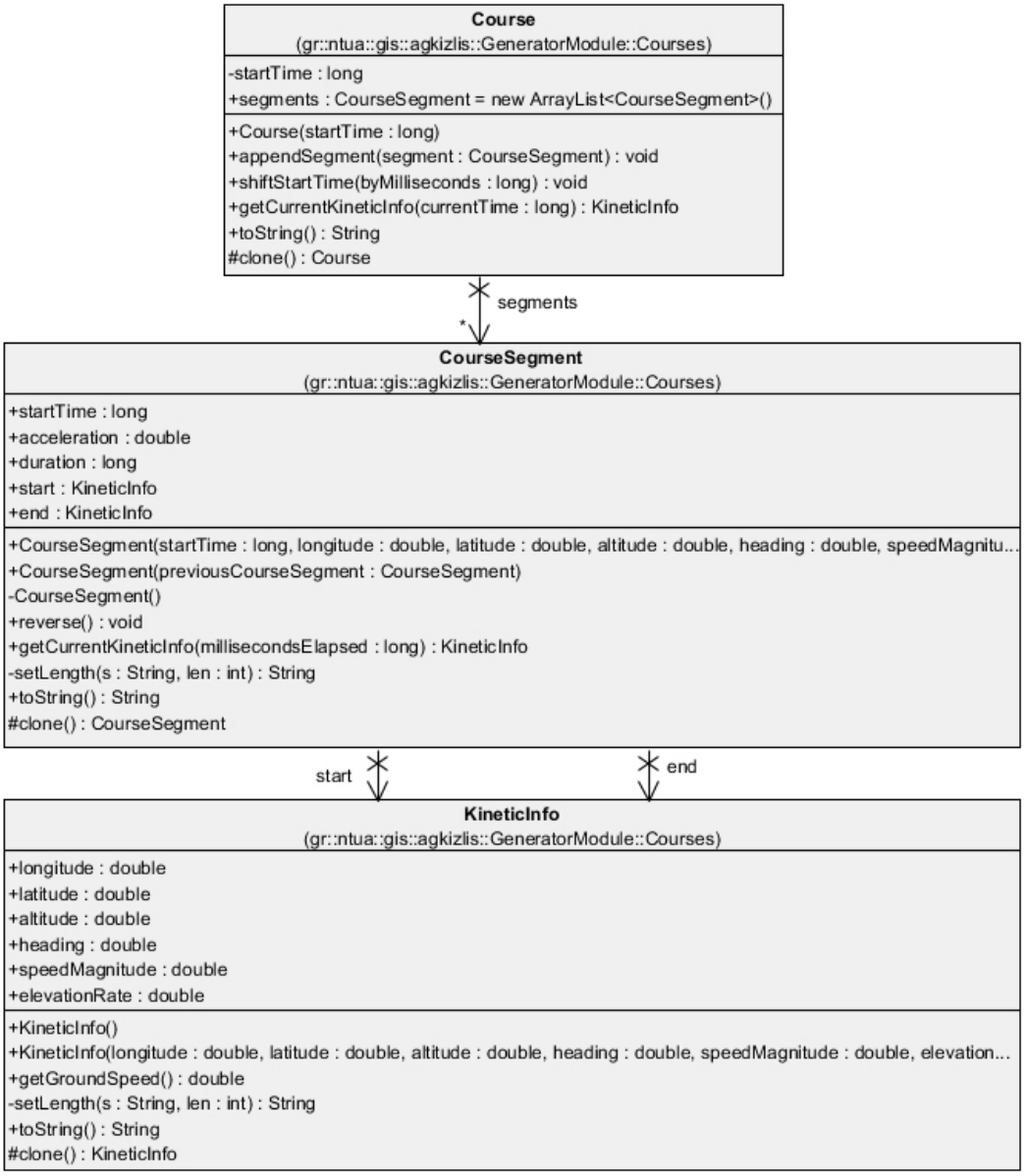
Σχήμα 15: Κλάσεις του Track generator settings

5.3.7 Generator module

Είναι ένα module με μία αποστολή, αρκετά πολύπλοκη ώστε να χρειάζονται αρκετές κλάσεις για να την πετύχει.

5.3.7.1 KineticInfo.java

Αυτή η κλάση (Σχήμα 16) απεικονίζει την κινητική κατάσταση ενός αντικειμένου σε μια δεδομένη στιγμή. Οι μεταβλητές της (με τη σειρά που εμφανίζονται στο σχήμα) είναι το γεωγραφικό μήκος και πλάτος, το ύψος, η διεύθυνση, το μέγεθος του ανύσματος της ταχύτητας και ο ρυθμός ανόδου (η κάθετη συνιστώσα του ανύσματος ταχύτητας). Η μέθοδος getGroundSpeed μας δίνει την οριζόντια συνιστώσα του ανύσματος ταχύτητας. Η setLength χρησιμοποιείται από την toString για την στοιχισμένη εμφάνιση του αντικειμένου ως αλφαριθμητικό, για σκοπούς αποσφαλμάτωσης.



Σχήμα 16: Κλάσεις του Generator module, μέρος 1^{ο}

5.3.7.2 CourseSegment.java

Ένα course segment είναι ένα ευθύγραμμο τμήμα της διαδρομής (course) ενός αεροσκάφους, που ξεκινάει κάποια συγκεκριμένη στιγμή (startTime), έχει σταθερή επιτάχυνση (acceleration) και καθορισμένη διάρκεια (duration). Ο χρόνος μετριέται πάντα σε milliseconds. Οι μεταβλητές start και end (Σχήμα 16) αποθηκεύουν την κινητική κατάσταση του αεροσκάφους στην αρχή και το τέλος του segment. Ο δεύτερος constructor δημιουργεί ένα segment που είναι συνέχεια του προηγούμενου, χρονικά και στο χώρο, διατηρώντας την ίδια επιτάχυνση, και διάρκεια μηδέν. Η μέθοδος reverse αντιστρέφει το segment, θέτοντας

την αρχή για τέλος, το τέλος για αρχή, αλλάζοντας κατά 180^{o} τη διεύθυνση της αρχής και του τέλους (η κινητική κατάσταση όπως είπαμε έχει και διεύθυνση και ταχύτητα), και αλλάζοντας το πρόσημο της επιτάχυνσης και του ρυθμού ανόδου στην αρχή και στο τέλος. Η μέθοδος getCurrentKineticInfo υπολογίζει τη θέση και την ταχύτητα του αεροσκάφους μετά από χρόνο millisecondsElapsed από την αρχή του segment. Για τον υπολογισμό της θέσης χρησιμοποιεί τα GeoTools. Και για να συγκρίνει την ταχύτητα με το μηδέν χρησιμοποιεί τις ρυθμίσεις που έχουμε ορίσει στο Global Settings.

5.3.7.3 Course.java

Ένα course (Σχήμα 16) αντιστοιχεί σε ολόκληρη τη διαδρομή που ακολουθεί ένα αεροσκάφος. Η μεταβλητή startTime περιέχει το χρόνο που το course ξεκινάει, και τα segments είναι τα τμήματα από τα οποία αποτελείται. Θεωρήθηκε περιττό να υλοποιηθούν καμπύλα τμήματα πτήσης, καθώς πρόκειται για έναν εξομοιωτή για σκοπούς ελέγχου, και όχι ρεαλιστικής αναπαράστασης της πραγματικότητας. Άλλωστε, αν χρειαστεί, μια καμπύλη μπορεί προσεγγιστικά να αναπαρασταθεί με ένα μεγάλο πλήθος μικρών ευθύγραμμων τμημάτων.

Η μέθοδος appendSegment προσθέτει ένα segment στο τέλος της σειράς των segment που ήδη υπάρχει. Η μέθοδος shiftStartTime προσθέτει την παράμετρο byMilliseconds στη μεταβλητή startTime και σε όλες τις μεταβλητές startTime όλων των segment που περιέχει. Η getCurrentKineticInfo εντοπίζει αρχικά σε ποιο segment βρισκόμαστε χρονικά στα currentTime milliseconds και μας επιστρέφει το αποτέλεσμα της getCurrentKineticInfo του segment αυτού.

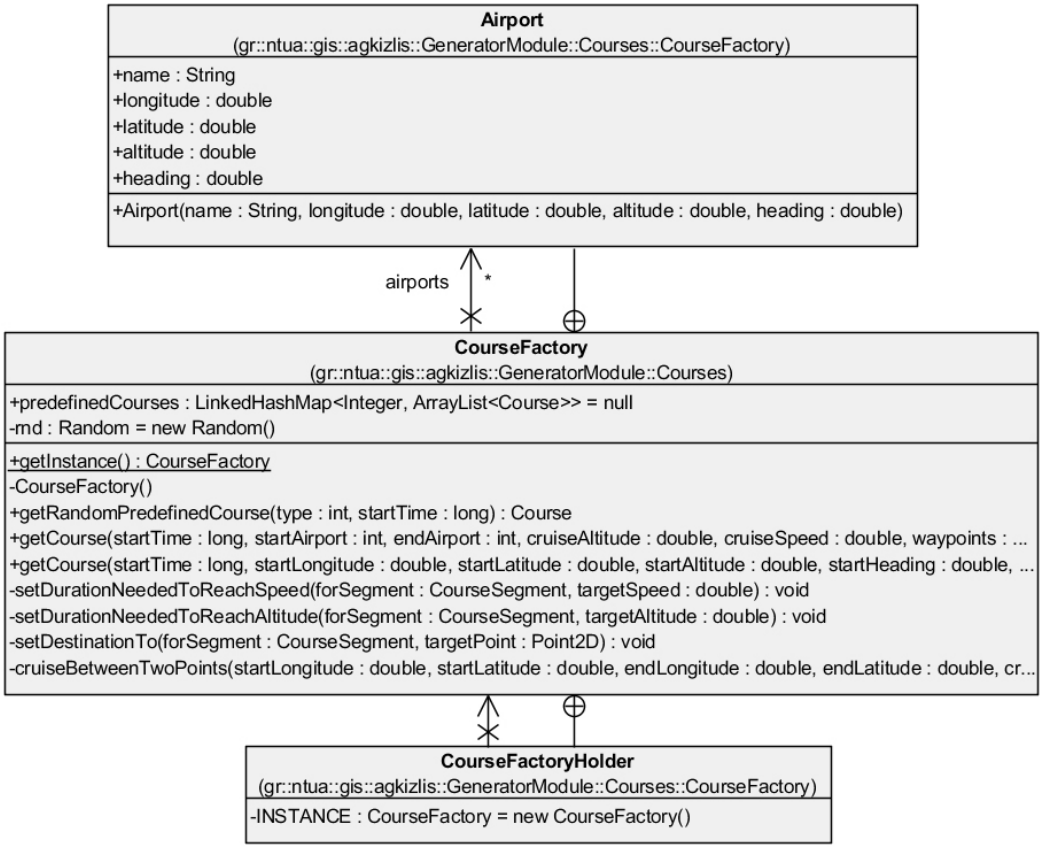
5.3.7.4 Airport.java

Ένα airport δεν είναι τίποτα άλλο παρά η απεικόνιση ενός αεροδρομίου. Έχει μια θέση στο χώρο, όνομα, και διεύθυνση. Αν θέλουμε να απεικονίσουμε και την αντίθετη διεύθυνση σε κάθε αεροδρόμιο, απλά δημιουργούμε άλλο ένα αεροδρόμιο με την αντίθετη διεύθυνση. Χρησιμοποιείται από το course factory για να δημιουργεί course που ξεκινάνε από κάπου και καταλήγουν κάπου.

5.3.7.5 CourseFactory.java

Το course factory (Σχήμα 17) είναι μια κλάση που δημιουργήθηκε έτσι ώστε εύκολα και γρήγορα να μπορούμε να δημιουργήσουμε course με ένα τυποποιημένο τρόπο. Είναι ένα singleton class. Οι τέσσερις τελευταίες μέθοδοι χρησιμοποιούνται από την getCourse για να αλλάξουν τα χαρακτηριστικά των segment που αποτελούν το course. Η getCourse φτιάχνει ένα καινούριο course και του προσθέτει τα κατάλληλα segments προκειμένου να απεικονίζει

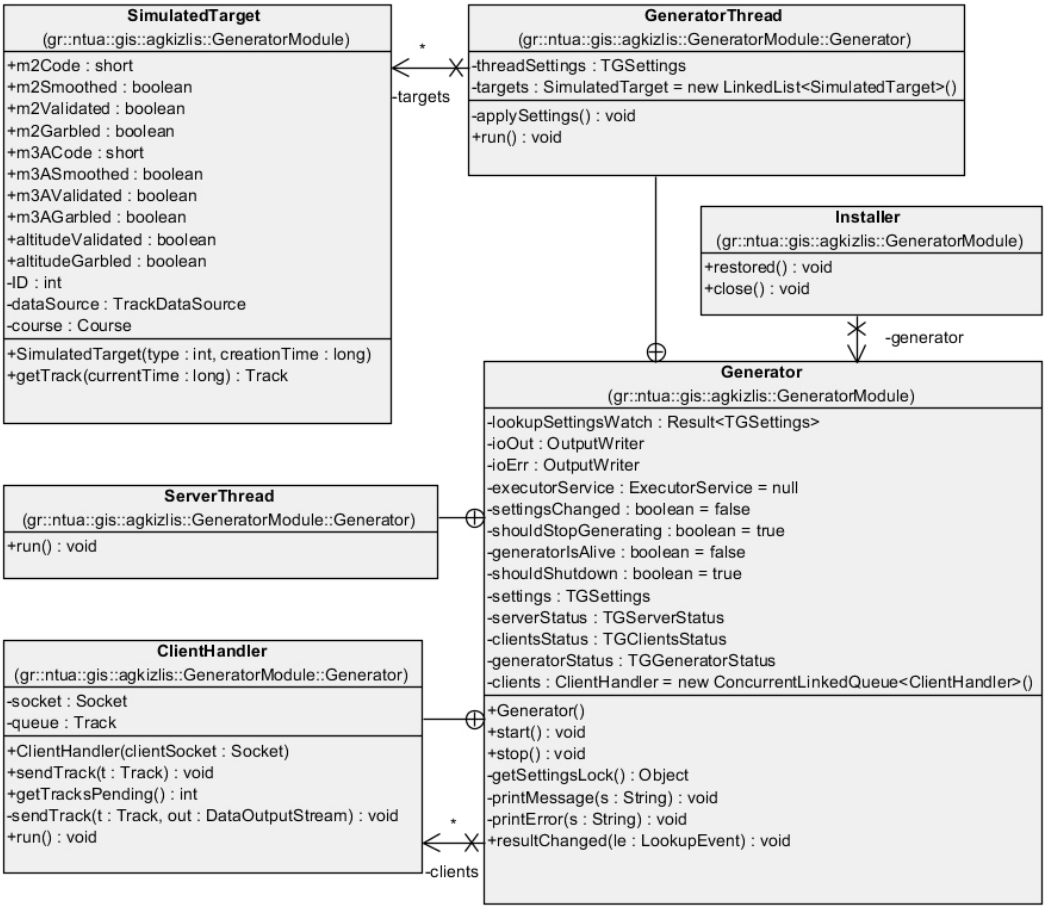
μια διαδρομή αεροσκάφους. Θέτουμε ένα σημείο εκκίνησης, ένα σημείο τερματισμού, ένα όριο ταχύτητας, ένα όριο ύψους και ένα σύνολο σημείων από όπου θα πρέπει να διέρχεται το αεροσκάφος (αν θέλουμε). Οι λεπτομέρειες της μεθόδου αναλύονται στο επόμενο κεφάλαιο.



Σχήμα 17: Κλάσεις του Generator module, μέρος 2ο

5.3.7.6 SimulatedTarget.java

Ένα simulated target είναι ένα αεροσκάφος. Προφανώς, σε μια πιο ρεαλιστική αναπαράσταση της πραγματικότητας, πολλά από τα πεδία του όπως φαίνονται στο Σχήμα 18, δε θα ήταν μέρος του αεροσκάφους, αλλά θα συμπληρώνονταν από το radar που θα τον ανίχνευε. Στην εξομοίωση που υλοποιήσαμε, δεν υπάρχει η αναπαράσταση ενός radar. Η γεννήτρια ρωτάει απευθείας το αεροσκάφος για τη θέση του. Έτσι, τα πεδία που θα συμπλήρωνε το radar, τα ορίζουμε στατικά μέσα στο αεροσκάφος ώστε να μπορεί να τα επιστρέψει στη γεννήτρια. Η σημαντική μέθοδος είναι η getTrack, η οποία δημιουργεί ένα νέο track, που αντιστοιχεί στο χρόνο που δίνουμε ως παράμετρο, και που τα πεδία του συμπληρώνονται από τα πεδία του simulated target και του kinetic info που επιστρέφει η getCurrentKineticInfo του course που περιέχει.



Σχήμα 18: Κλάσεις του Generator module, μέρος 3^{ο}

5.3.7.7 Generator.java

Αυτή η κλάση είναι ο πυρήνας της γεννήτριας (Σχήμα 18). Η μεταβλητή lookupSettingsWatch παρακολουθεί το Global lookup για αλλαγές στα instance της κλάσης TGSettings που έχουν αναρτηθεί σε αυτό. Οι ioOut και ioErr φροντίζουν για την εμφάνιση μηνυμάτων στο παράθυρο Output της πλατφόρμας, για σκοπούς αποσφαλμάτωσης. Η executorService μας βοηθά να ξεκινάμε καινούρια threads όποτε απαιτείται. Ακολουθούν flags για την επικοινωνία με τα threads που δημιουργεί, μεταβλητές για την κατάσταση και τις ρυθμίσεις της γεννήτριας και τέλος, μια λίστα από τους λήπτες (clients) των tracks που παράγονται.

Η εσωτερική κλάση GeneratorThread τρέχει σε ξεχωριστό thread και παράγει συνεχώς tracks σύμφωνα με τις ρυθμίσεις, τα οποία και προωθεί σε κάθε ClientHandler που είναι στη λίστα clients. Η εσωτερική κλάση ServerThread τρλέχει σε ξεχωριστό thread, «ακούει» σε μια θύρα για αιτήσεις σύνδεσης από clients και για κάθε έναν που θέλει να συνδεθεί, δημιουργεί ένα instance ClientHandler, το προσθέτει στη λίστα clients, και δημιουργεί ένα καινούριο thread

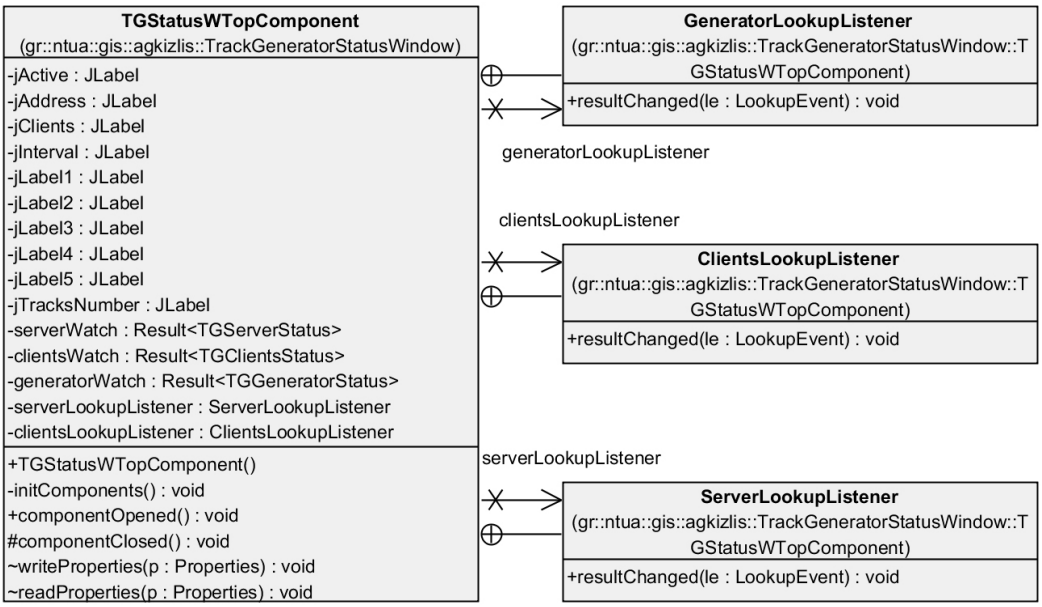
όπου το instance θα εκτελείται. Η εσωτερική κλάση ClientHandler προωθεί μέσω δικτύου στον δέκτη που αντιστοιχεί σε αυτή, τα tracks που το GeneratorThread έχει παράγει και της έχει προωθήσει. Η επικοινωνία δε γίνεται μέσω του πρωτοκόλλου http, αλλά με TCP sockets, όπως γίνεται στα πραγματικά δίκτυα διαμοιρασμού ιχνών αεροσκαφών.

Από τις μεθόδους, η start και stop εκκινούν και τερματίζουν τη γεννήτρια, και η resultChanged καλείται από το Global lookup όταν υπάρχει αλλαγή στα instance της κλάσης TGSettings που είναι αναρτημένα σε αυτό.

5.3.7.8 Installer.java

Ο installer υλοποιεί δύο από τις μεθόδους που μπορεί να υλοποιήσει ένας installer, αυτή που αντιστοιχεί στο άνοιγμα του module από την πλατφόρμα (restored), και αυτή που αντιστοιχεί στο κλείσιμό του (close). Στη μία δημιουργείται ένα instance της κλάσης Generator και καλείται η μέθοδος start, και στην άλλη καλείται η μέθοδος stop του Generator που είχαμε δημιουργήσει.

Κάθε φορά που έχουμε υλοποιήσει έναν installer, έχουμε ακριβώς το ίδιο σενάριο. Δημιουργούμε ένα αντικείμενο και ξεκινάμε την εκτέλεσή του, ώστε να μας δίνει τη λειτουργικότητα που θέλουμε. Αυτό όπως είπαμε, συμβαίνει γιατί μόνο οι κλάσεις που εμφανίζουν κάποιο παράθυρο εκτελούνται αυτόματα από την πλατφόρμα. Οι υπόλοιπες πρέπει να ξεκινήσουν χειροκίνητα, όποτε εμείς καλέσουμε τον κώδικα. Αν θέλουμε λοιπόν ο κώδικας να εκτελείται με την εκκίνηση του module, χρειαζόμαστε έναν installer.



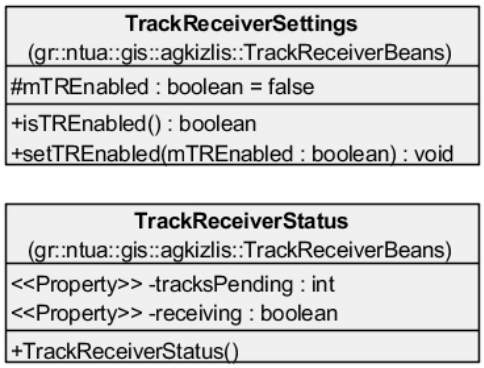
Σχήμα 19: Κλάσεις του Track generator status module

5.3.8 Track generator status

Αυτό το module υλοποιεί το παράθυρο όπου βλέπουμε την κατάσταση της γεννήτριας. Όπως φαίνεται στο Σχήμα 19, έχει τρεις εσωτερικές κλάσεις, μία για κάθε είδος κατάστασης που παρουσιάζει (κατάσταση την γεννήτριας, του εξυπηρετητή και των δεκτών που είναι συνδεδεμένοι), οι οποίες εμφανίζουν στο παράθυρο τις πληροφορίες κατάστασης, όποτε ενημερώνονται από το Global lookup ότι υπάρχουν αλλαγές. Έχουμε ήδη παρουσιάσει τη διαδικασία με την οποία αυτό γίνεται, και δε χρειάζεται περαιτέρω ανάλυση.

5.3.9 Track receiver beans

Αυτό το module (Σχήμα 20) αποτελεί μέρος του Δέκτη της Ανεξάρτητης Μονάδας Εισόδου που υλοποιήσαμε. Όπως και το Track generator beans module, αποτελεί έναν ενδιάμεσο κρίκο ανάμεσα στο Track receiver και Track receiver window έτσι ώστε να τα αποσυνδέει. Η κλάση TrackReceiverSettings έχει μόνο μία ρύθμιση (αν ο δέκτης θα λαμβάνει ίχνη ή όχι) και η κλάση TrackReceiverStatus περιέχει πεδία για το αν ο δέκτης λαμβάνει ίχνη, και πόσα ίχνη έχει λάβει αλλά δεν έχουν προωθηθεί στον Αποστολέα (δηλαδή ο αριθμός των ιχνών που περιέχει το εσωτερικό του buffer).



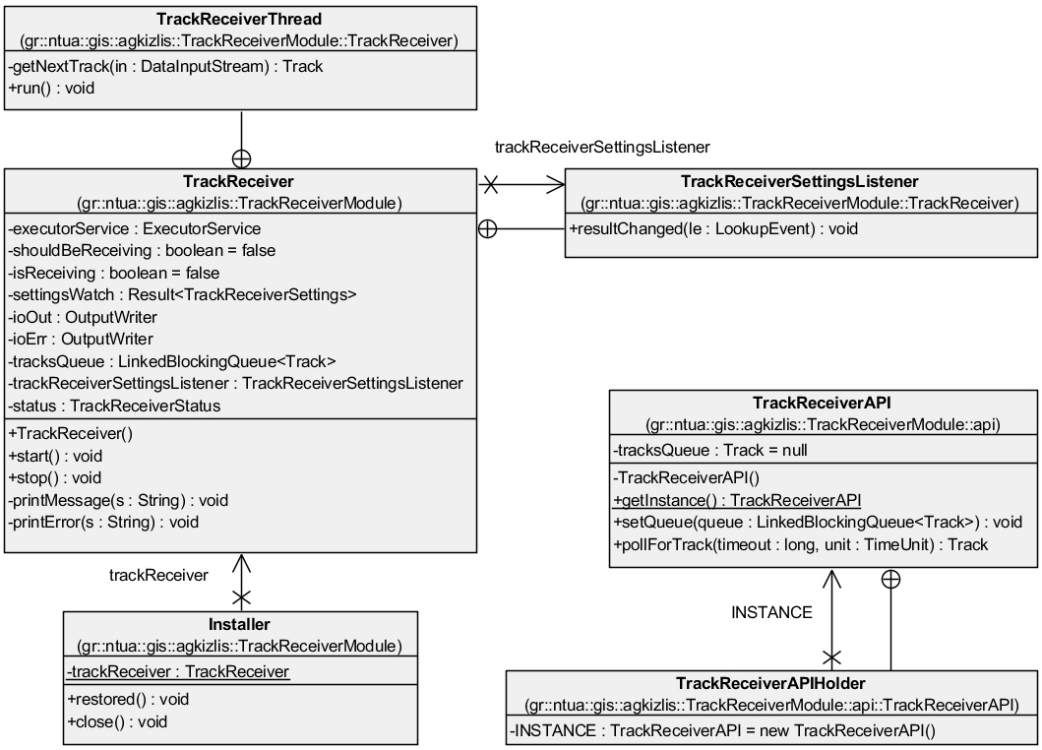
Σχήμα 20: Κλάσεις του Track receiver beans module

5.3.10 Track receiver

Σε αυτό το module υλοποιείται ο Δέκτης στο Σχήμα 5.

5.3.10.1 TrackReceiverAPI

Όπως είπαμε, ο Δέκτης συνδέεται στενά με τον Αποστολέα. Προκειμένου να «χαλαρώσουμε» αυτή τη σύνδεση, υπάρχει μια κλάση που λειτουργεί ως ενδιάμεσος, η TrackReceiverAPI (Σχήμα 21), στην οποία συνδέεται η κλάση Trackeceiver δίνοντας μια σύνδεση στο εσωτερικό του buffer, και ο Αποστολέας, τραβώντας ίχνη από αυτό το buffer. Είναι μια singleton class.



Σχήμα 21: Κλάσεις του Track receiver module

5.3.10.2 TrackReceiver

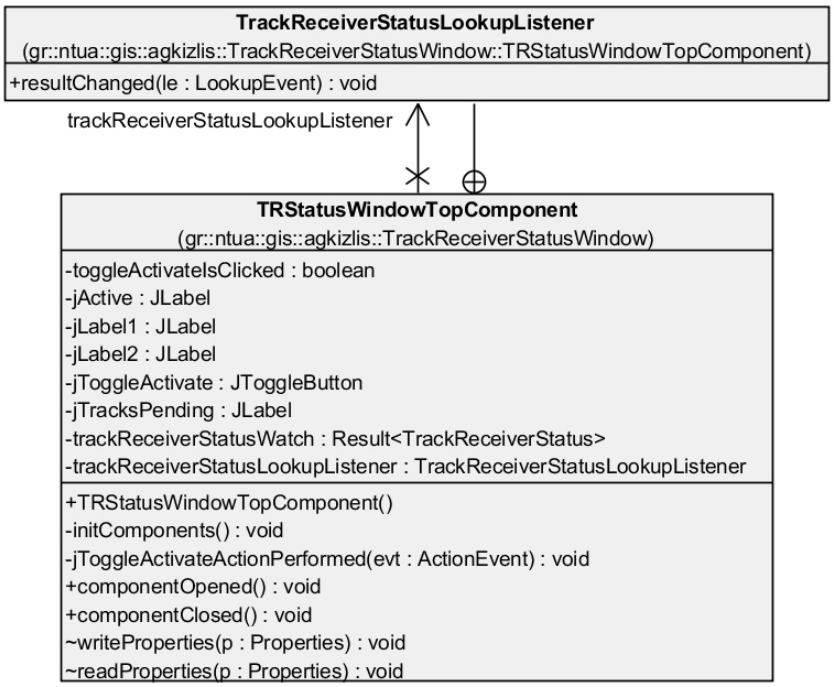
Η κλάση αυτή (Σχήμα 21) υλοποιεί τον ίδιο το Δέκτη. Η εσωτερική κλάση TrackReceiverSettingsListener υπάρχει για να δέχεται από το Global lookup τις ρυθμίσεις για τον Δέκτη. Από την εσωτερική κλάση TrackReceiverThread δημιουργείται ένα instance για να δέχεται ίχνη από την εξωτερική πηγή, μέσω ενός TCP socket, σε ξεχωριστό thread.

Η μεταβλητή executorService υπάρχει για να ξεκινάει ένα νέο thread όπου θα τρέχει το instance του TrackReceiverThread. Ακολουθούν δύο flags για την επικοινωνία με το thread, η μεταβλητή settingsWatch που υλοποιεί τη σύνδεση με το Global lookup, οι ioOut και ioErr για έξοδο στο παράθυρο Output για σκοπούς αποσφαλμάτωσης, η tracksQueue που είναι το buffer των ιχνών που λαμβάνονται, η tracksReceiverSettingsListener για να ενημερώνεται για τις αλλαγές των ρυθμίσεων στο Global lookup, και η status που κρατά πληροφορίες για την κατάσταση του δέκτη. Οι μέθοδοι start και stop έχουν την ίδια λειτουργικότητα που περιγράψαμε στην κλάση Generator.

5.3.11 Track receiver window

Αυτό το module είναι κάτι αντίστοιχο με μια συγχώνευση των module Track generator settings και Track generator status, για τον δέκτη. Είναι μόνο μια ρύθμιση και δύο

μεταβλητές κατάστασης που πρέπει να προβληθούν, επομένως η δημιουργία δύο παραθύρων για αυτό το σκοπό ήταν περιττή. Στο Σχήμα 22 φαίνονται οι κλάσεις του module, που δεν είναι παρά η κλάση που υλοποιεί το παράθυρο, και μια εσωτερική κλάση που παρακολουθεί το Global lookup για αλλαγές στην κατάσταση του δέκτη.

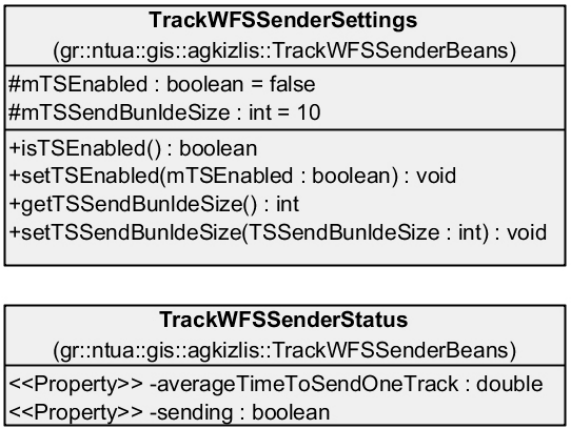


Σχήμα 22: Κλάσεις του Track receiver window module

5.3.12 Track WFS sender beans

Αυτό το module αποτελεί το συνδετικό κρίκο ανάμεσα στο module Track WFS sender και τα modules Track WFS sender settings και Track WFS sender status (Σχήμα 23).

Περιέχει δύο κλάσεις, μία για τις ρυθμίσεις και μία για την κατάσταση του δέκτη. Το μοντέλο χρήσης των κλάσεων αυτών έχει περιγραφεί παραπάνω με λεπτομέρεια.



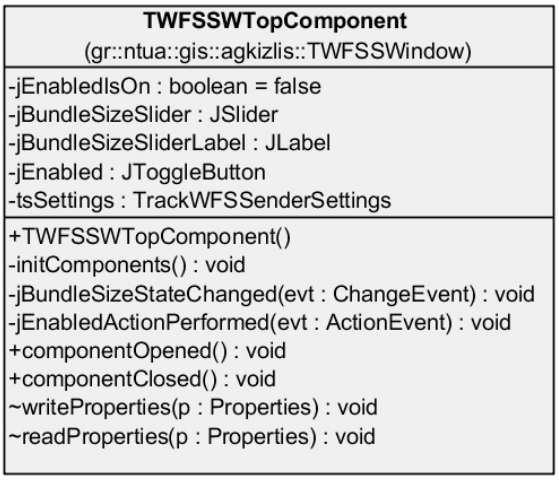
Σχήμα 23: Κλάσεις του Track WFS sender beans

Οι ρυθμίσεις που υπάρχουν είναι δύο, η ενεργοποίηση του αποστολέα, και το πόσα ίχνη θα στέλνει ταυτόχρονα στον GeoServer με κάθε WFS insert request. Υλοποιήθηκε ως ρύθμιση γιατί επηρεάζει την απόδοση του

συστήματος, και έπρεπε να ελεγχθεί ποια τιμή είναι καλύτερη. Τα αποτελέσματα μάλιστα όπως θα δούμε στο κεφάλαιο 7, ήταν απροσδόκητα.

5.3.13 Track WFS sender settings

Πρόκειται για ένα module με μία κλάση που εμφανίζει ένα παράθυρο με δύο ρυθμιστικά, ένα για κάθε ρύθμιση που υπάρχει για τον Αποστολέα. Παρουσιάζεται στο Σχήμα 24.

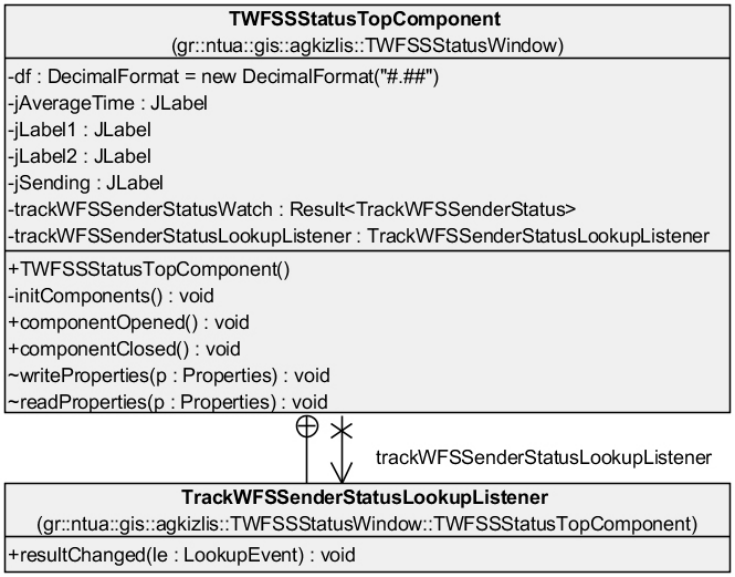


Σχήμα 24: Κλάσεις του Track WFS settings

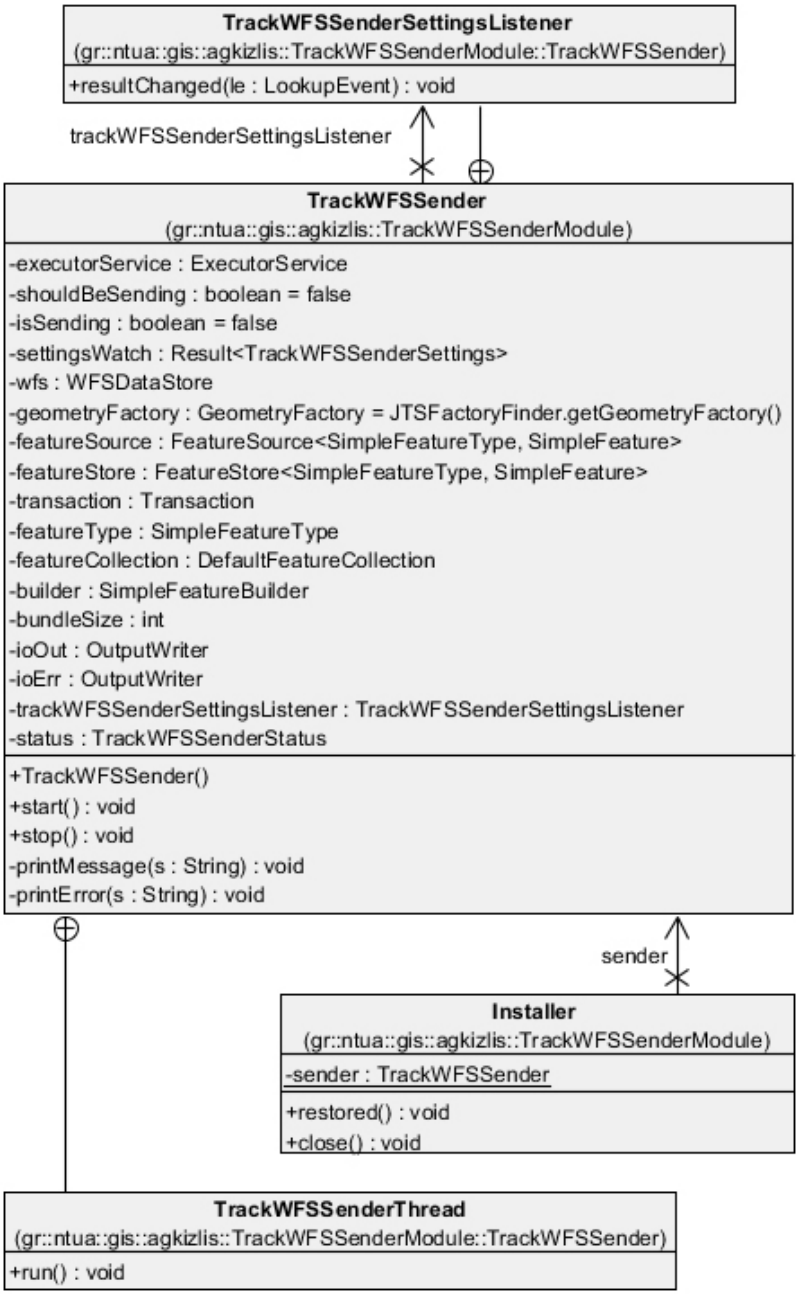
5.3.14 Track WFS sender

Σε αυτό το module (Σχήμα 26) υπάρχουν τέσσερις κλάσεις. Τις τρεις από αυτές τις έχουμε δει επανειλημμένα, τον installer για την εκκίνηση, την κλάση TrackWFSSettingsListener για την παρακολούθηση του Global lookup και την TrackWFSSenderThread που εκτελείται σε ξεχωριστό thread για να στέλνει τα ίχνη στον GeoServer.

Από τις μεταβλητές της κλάσης TrackWFSSender, βλέπουμε κάποιες που έχουμε συναντήσει και σε άλλα module, και πολλές ακόμα που χρησιμοποιούνται για τη χρήση του πρωτοκόλλου WFS και τη δημιουργία της λίστας των ιχνών που θα σταλούν με WFS insert query στον GeoServer. Στο επόμενο κεφάλαιο θα δούμε πιο αναλυτικά αυτή τη διαδικασία.



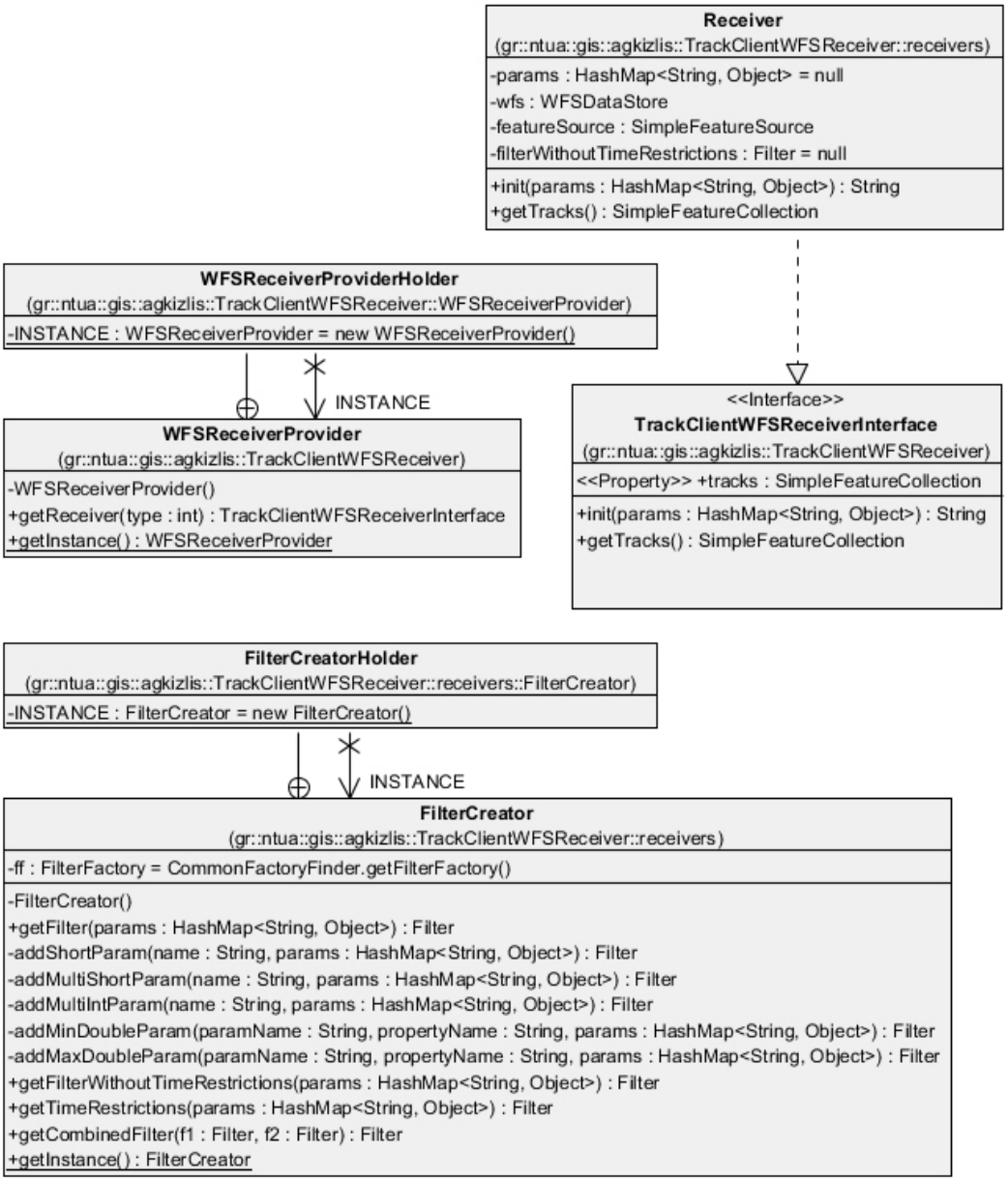
Σχήμα 25: Κλάσεις του Track WFS sender status module



Σχήμα 26: Κλάσεις του Track WFS sender module

5.3.15 Track WFS sender status

Περιέχει μία μόνο κλάση (Σχήμα 25), η οποία υλοποιεί ένα παράθυρο όπου παρουσιάζεται η κατάσταση του αποστολέα.



Σχήμα 27: Κλάσεις του Track client WFS receiver module

5.3.16 Track client WFS receiver

Στο Σχήμα 27 μπορούμε να δούμε τις κλάσεις αυτού του module. Όπως θα δούμε στην παράγραφο 5.4, ο GeoServer ρυθμίζεται ώστε να δίνει μια εικόνα των δεδομένων της γεωχωρικής βάσης δεδομένων, όπως εμείς την επιθυμούμε, μέσω ενός query. Μπορεί μάλιστα να έχει πολλές εικόνες για τα δεδομένα, με διαφορετικές πληροφορίες το καθένα. Έχουμε υλοποιήσει μια εικόνα όπου για ένα παράθυρο χρόνου μας δίνει το πιο πρόσφατο ίχνος κάθε αεροσκάφους. Έχουμε υλοποιήσει επίσης μια εικόνα όπου παίρνουμε όλα τα ίχνη κάθε αεροσκάφους μέσα σε ένα παράθυρο χρόνου, παρόλο που δεν την αξιοποιούμε πουθενά

στην εφαρμογή. Για να υπάρχει όμως η δυνατότητα εύκολης αξιοποίησης μελλοντικά (πράγμα που κάνει την εφαρμογή εύκολα επεκτάσιμη) έχουμε υιοθετήσει μια δομή σε αυτό το module που μας επιτρέπει να επιστρέφουμε σε όποιον ζητήσει (πχ το παράθυρο παρουσίασης) ένα δέκτη με βάση μια παράμετρο. Ανάλογα την τιμή της παραμέτρου, επιστρέφεται ένας δέκτης που ζητάει δεδομένα μιας διαφορετικής κάθε φορά εικόνας. Η υλοποίηση εδώ έχει μόνο ένα δέκτη, που αξιοποιεί μόνο τη μία εικόνα, αλλά είναι πολύ εύκολο να δημιουργηθεί και ένας δεύτερος για μια δεύτερη εικόνα.

5.3.16.1 WFSReceiverProvider.java

Αυτή η κλάση είναι που φαίνεται εξωτερικά στο module. Είναι ένα singleton class που έχει ουσιαστικά μία μέθοδο, την getReceiver, η οποία μας επιστρέφει έναν receiver που υλοποιεί το interface TrackClientWFSReceiverInterface. Με τη χρήση του interface, δε χρειάζεται ο κλάση που θα χρησιμοποιήσει το receiver να γνωρίζει λεπτομέρειες για την υλοποίησή του.

5.3.16.2 TrackClientWFSReceiverInterface.java

Το interface που θα πρέπει να υλοποιεί κάθε δέκτης. Έχει μόνο δύο μεθόδους, μια για την αρχικοποίηση του δέκτη βάση κάποιων παραμέτρων, και για να μας δώσει τα ίχνη. Τα ίχνη επιστρέφονται σε μορφή SimpleFeatureCollection, που είναι μια δομή των GeoTools για την αναπαράσταση γεωχωρικών δεδομένων.

5.3.16.3 Receiver.java

Ο μοναδικός δέκτης της υλοποίησής μας είναι αυτή η κλάση. Αρχικοποιείται σύμφωνα με τις παραμέτρους της μεθόδου init, χρησιμοποιεί την κλάση FilterCreator για να δημιουργήσει ένα φίλτρο με όσους περιοριστικούς παράγοντες εισάγουμε μέσω του παράθυρου παρουσίασης, και επιστρέφει τα ίχνη που ζητάει από τον GeoServer με τη μέθοδο getTracks.

Το φίλτρο που χρησιμοποιείται για τα δεδομένα που θα επιστραφούν έχει δύο μέρη. Ένα είναι με περιορισμούς που δεν έχουν σχέση με το χρόνο, και το άλλο με χρονικούς περιορισμούς. Αυτό συμβαίνει ώστε μόνο το φίλτρο των χρονικών περιορισμών (και όχι ολόκληρο το φίλτρο από την αρχή) να δημιουργείται κάθε φορά που καλούμε την getTracks. Έτσι, αν θέλουμε να μας επιστρέφει τα πιο πρόσφατα δεδομένα κάθε φορά, η getTracks θα «μετακινεί» το παράθυρο χρόνου κάθε φορά έτσι ώστε το τέλος του να είναι τη στιγμή που ζητάμε τα ίχνη.

5.3.16.4 FilterCreator.java

Πρόκειται για μια singleton class που έχει thread safe μεθόδους για τη δημιουργία OGC φίλτρων που θα στέλνονται από τους receivers προς τον GeoServer. Τα ονόματα των public μεθόδων αποσαφηνίζουν αρκετά τη λειτουργία της καθεμίας. Ο receiver περνάει ως παράμετρο τις παραμέτρους που του δόθηκαν κατά την αρχικοποίησή του σε κάποια από αυτές τις μεθόδους, και η μέθοδος ανιχνεύει μέσα σε αυτές τις παραμέτρους όσες αφορούν περιορισμούς στα δεδομένα που πρέπει να επιστραφούν (πχ "TrackID" : List {1045, 7895}) και δημιουργεί ένα φίλτρο με βάση αυτές.

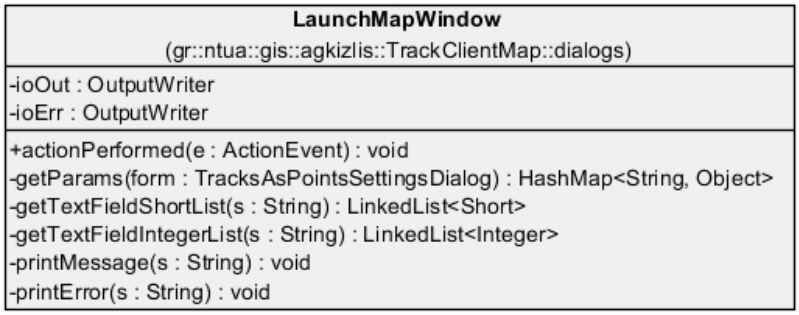
5.3.17 JCalendarWrapper

Όπως είδαμε στα GeoTools, έτσι κι εδώ αυτό το module δεν περιέχει καθόλου κώδικα. Υπάρχει μόνο για να κάνει διαθέσιμη στα υπόλοιπα modules τη βιβλιοθήκη JCalendar.jar που περιέχει ένα component για την επιλογή ημερομηνίας, μια λειτουργικότητα που λείπει από τα Swing components που είναι διαθέσιμα μέσω της τυπικής εγκατάστασης του NetBeans Platform. Χρησιμοποιείται από το module Track client map για την εισαγωγή περιορισμού στο παράθυρο χρόνου.

5.3.18 Track client map

Όπως έχουμε ήδη πει, πρόκειται για το Παράθυρο Παρουσίασης και το Υποσύστημα Εισαγωγής Φίλτρων μαζί. Θα δούμε στη συνέχεια τις κλάσεις του module.

5.3.18.1 LaunchMapWindow.java



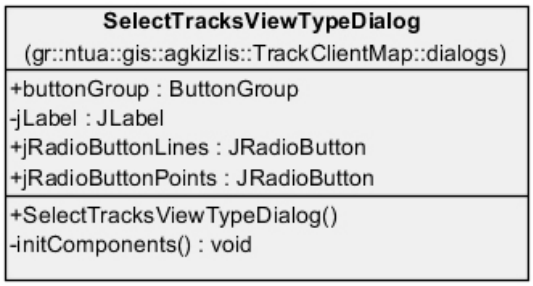
Σχήμα 28: Κλάση LaunchMapWindow του module Track client map

Στο Σχήμα 28 βλέπουμε την κλάση LaunchMapWindow. Η κλάση αυτή υλοποιεί μια εντολή στο μενού της εφαρμογής. Η μέθοδος που καλείται είναι η actionPerformed, με την οποία εμφανίζεται ένα παράθυρο διαλόγου που μας επιτρέπει να επιλέξουμε τον τύπο του ερωτήματος που θα κάνουμε (όπως είπαμε έχουμε υλοποιήσει μόνο ένα τύπο), στη συνέχεια

ανάλογα με την επιλογή μας εμφανίζει ένα μήνυμα ή ανοίγει ένα διάλογο όπου εισάγουμε τα φίλτρα για τα δεδομένα που θέλουμε. Τέλος, όταν τελειώσουμε με την εισαγωγή των φίλτρων, δημιουργεί μια λίστα παραμέτρων από όλες τις επιλογές μας και ανοίγει ένα παράθυρο παρουσίασης σύμφωνο με αυτές τις επιλογές.

5.3.18.2 SelectTrackViewTypesDialog.java

Αυτή η κλάση (Σχήμα 29) υλοποιεί ένα διάλογο όπου εισάγουμε τον τύπο των αποτελεσμάτων που θέλουμε. Μας δίνει δύο επιλογές, αλλά μόνο η πρώτη είναι ενεργή. Η δεύτερη υπάρχει ως επίδειξη του πόσο εύκολη είναι η υλοποίηση εναλλακτικών προβολών.



Σχήμα 29: Κλάση SelectTrackViewTypesDialog.java του module Track client map

5.3.18.3 TracksAsPointsSettingsDialog

Ο επόμενος διάλογος που εμφανίζεται υλοποιείται από αυτή την κλάση (Σχήμα 30). Μας επιτρέπει να καθορίσουμε τους περιορισμούς για τα ίχνη που θα προβληθούν. Με βάση τις τιμές που θα εισάγουμε εδώ, δημιουργείται από το LaunchMapWindow ένα HashMap από παραμέτρους, με τις οποίες αρχικοποιείται ο receiver και δημιουργούνται τα φίλτρα από τον FilterCreator.

5.3.18.4 TrackClientMapWindowHelper.java

Η κλάση αυτή (Σχήμα 31) περιέχει όλη τη λειτουργικότητα του παράθυρου παρουσίασης, ώστε η κλάση που δείχνει τα αποτελέσματα στην οθόνη να είναι πιο «καθαρή» και ευανάγνωστη. Είναι μια singleton class.

Η εσωτερική κλάση TimeColorEntry αντιστοιχεί ένα χρώμα σε ένα χρονικό όριο. Θα δούμε στη συνέχεια πως χρησιμοποιείται.

Η μέθοδος setupToolbar μας επιστρέφει μια μπάρα εργαλείων (ζουμ, μετατόπιση, επιλογή κτλ) που θα εμφανίσουμε στο παράθυρο του χάρτη. Σε αυτό το σημείο να αναφέρουμε πως αν δει κανείς τον κώδικα, υπάρχουν αρκετές κλάσεις στα πακέτα action και tool μέσα στο πακέτο gr.ntua.gis.agkizlis.TrackClientMap.swing, οι οποίες όμως είναι σχεδόν πανομοιότυπες με τις πρωτότυπες των GeoTools. Προσφέρουν τη λειτουργικότητας της μπάρας εργαλείων. Όμως τα GeoTools χρησιμοποιούσαν εικονίδια μεγέθους 24x24 pixels και για το εικονίδιο του κάθε εργαλείου, αλλά και για το εικονίδιο του δρομέα όταν ένα από τα εργαλεία είχε επιλεγεί. Η διάσταση του δρομέα στα windows είναι 32x32 pixels, με αποτέλεσμα να φαίνονται πολύ άσχημα. Έτσι δημιουργήσαμε μια σειρά από εικονίδια 32x32

pixels για να απεικονίζουν το δρομέα κάθε εργαλείου, και αντιγράψαμε τις κλάσεις εδώ έτσι ώστε με μια μικρή τροποποίηση να χρησιμοποιούν τα νέα εικονίδια για το δρομέα.

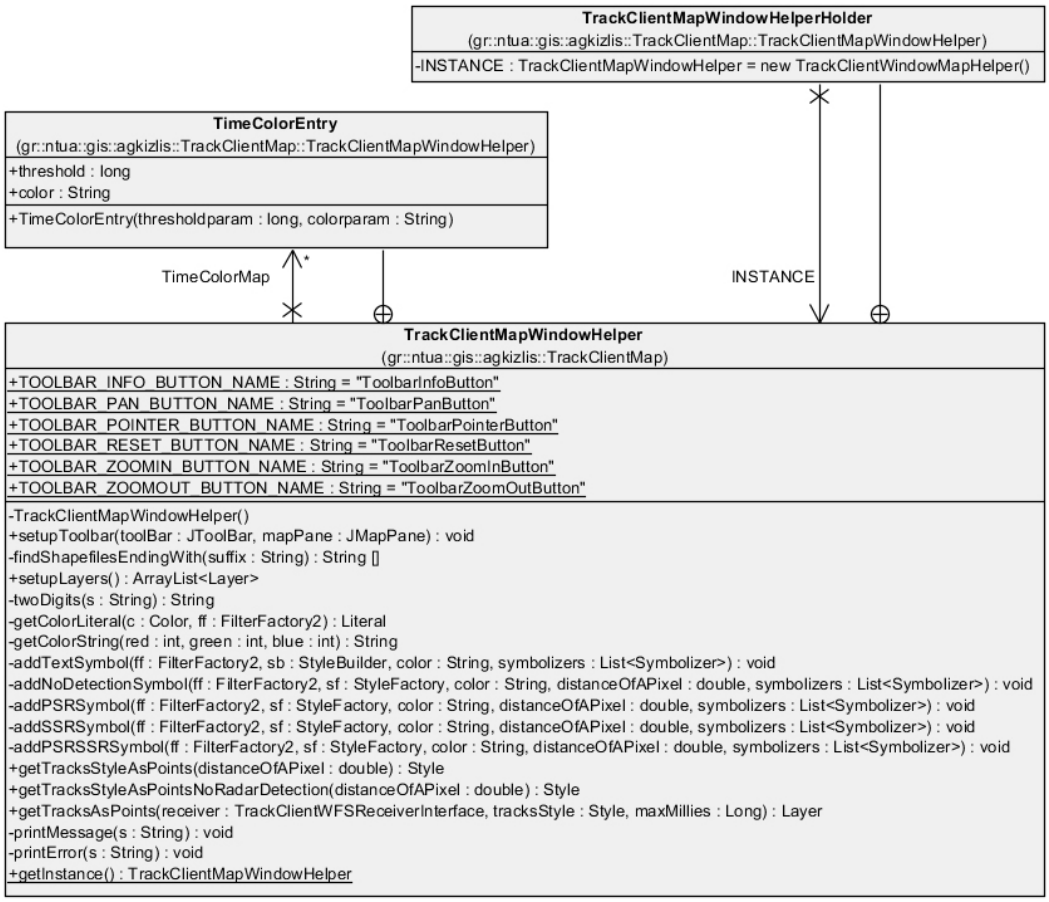
Η μέθοδος setupLayers διαβάζει από το δίσκο όσα αρχεία shapefile βρει διαθέσιμα (με συγκεκριμένη μορφή ονόματος) και δημιουργεί από ένα layer για το καθένα.

Η μέθοδος getTracksStyleAsPoints επιστρέφει το στυλ με το οποίο θα απεικονιστούν τα ίχνη που θα επιστραφούν από τον GeoServer. Εδώ χρησιμοποιείται και η εσωτερική κλάση TimeColorEntry. Έχουμε ορίσει ότι ανάλογα με την ηλικία κάθε ίχνους (σε seconds) θα απεικονίζεται και με διαφορετικό χρώμα. Έτσι έχουμε μια λίστα με αντιστοιχίσεις χρόνου-χρώματος, που χρησιμοποιούμε για να υλοποιήσουμε το χρωματικό κώδικα.

Η μέθοδος getTracksAsPoints συνθέτει όλα όσα είδαμε για να μας επιστρέψει το layer με τα ίχνη. Εκτελεί το ερώτημα στον GeoServer, παίρνει την απάντηση, δημιουργεί μια νέα συλλογή για ίχνη και αντιγράφει την απάντηση σε αυτή, κάνοντας τρεις ρυθμίσεις κατά την αντιγραφή. Θέτει την ηλικία των ιχνών που λάβαμε στη σωστή τιμή, και στρογγυλοποιεί το

Σχήμα 30: Κλάση TracksAsPointsSettingsDialog.java

ύψος και την ταχύτητα, ώστε να μην προβληθούν με πολλά δεκαδικά στοιχεία στην οθόνη. Μετά χρησιμοποιεί το αντίγραφο και το στυλ που επιστρέφει η getTracksStyleAsPoints για να συνθέσει ένα νέο layer το οποίο και μας επιστρέφει.



Σχήμα 31: Κλάση TrackClientMapWindowHelper.java του module Track client map

5.3.18.5 TrackClientMapWindow

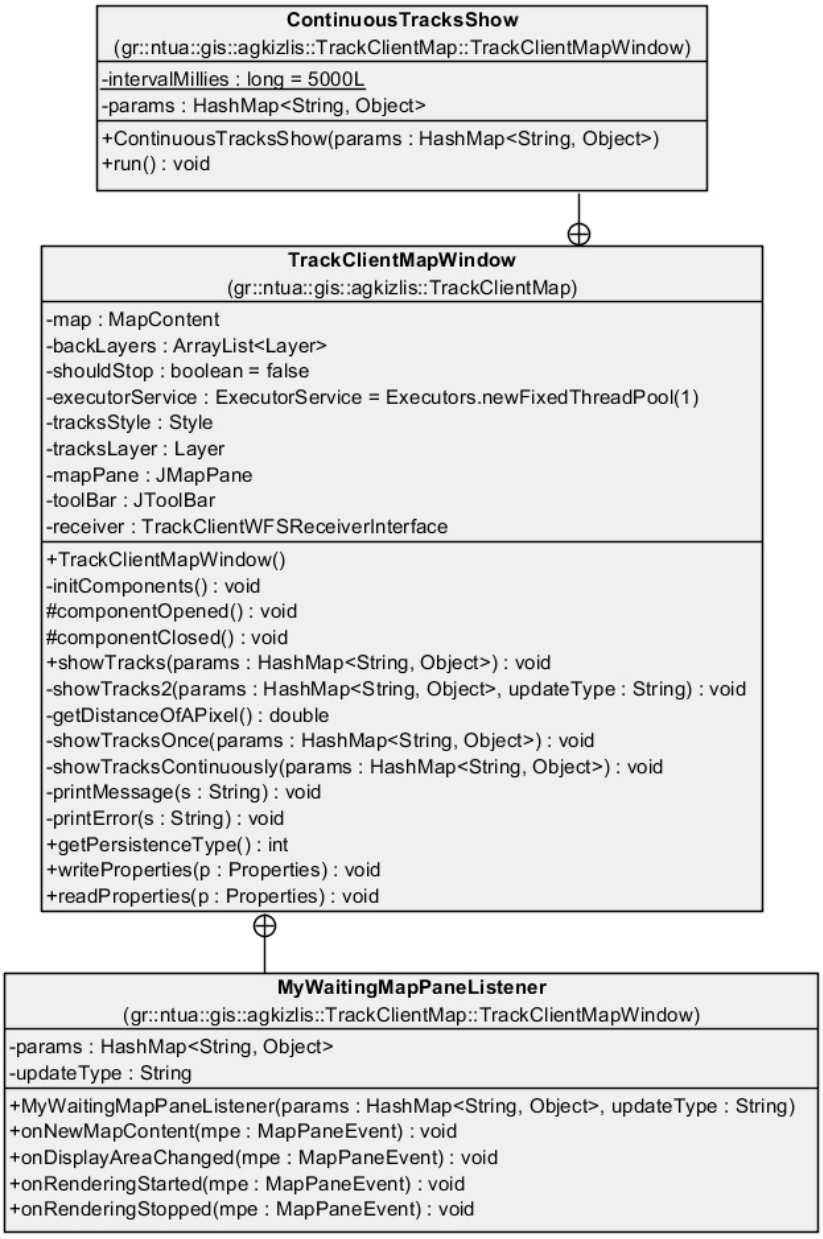
Το LaunchMapWindow παίρνει τις παραμέτρους από το χρήστη, δημιουργεί ένα instance του TrackClientMapWindow (Σχήμα 32), το εμφανίζει στην οθόνη, και καλεί τη μέθοδο showTracks.

Η μέθοδος showTracks ζητάει το σωστό receiver (υπάρχει μόνο ένας), τον αρχικοποιεί ανάλογα με τις παραμέτρους, εμφανίζει το toolbar, δημιουργεί ένα χάρτη, εισάγει σε αυτόν τα layers που αποτελούν το υπόβαθρο και προέρχονται από shapefiles και περιμένει να εμφανιστούν στην οθόνη.

Η κλάση MyWaitingPaneListener υλοποιεί ένα interface που ορίζεται στα GeoTools και έχει μεθόδους που καλούνται από τον renderer του χάρτη σε ορισμένα γεγονότα. Χρησιμοποιούμε

αυτή τη κλάση για να ενημερωθούμε από τον renderer για το πότε τελείωσε να δείχνει το υπόβαθρο.

Όταν αυτό συμβεί, καλείται η showTracks2 (που αποτελεί ουσιαστικά συνέχεια της showTracks) και ανάλογα με το τι ζήτησε ο χρήστης, κάνει ένα από τα δύο ακόλουθα. Αν ο χρήστης ζήτησε να δει την αεροπορική εικόνα σε κάποια στιγμή στο παρελθόν, καλεί την showTracksOnce και την εμφανίζει. Αν ζήτησε νε βλέπει την παρούσα αεροπορική εικόνα, καλεί την showTracksContinuously, η οποία δημιουργεί ένα νέο thread στο οποίο εκτελεί ένα instance της κλάσης ContinuousTracksShow.



Σχήμα 32: Κλάση TrackClientMapWindow του module Track client map

Η κλάση ContinuousTracksShow ανά πέντε δευτερόλεπτα εκτελεί την εξής ακολουθία: αφαιρεί από το χάρτη το layer των ιχνών (αν υπάρχει), ζητάει από την κλάση TrackClientMapWindowHelper το νέο layer ιχνών, και το προσθέτει στο χάρτη. Έτσι έχουμε συνεχή (ανά πέντε δευτερόλεπτα) ανανέωση της αεροπορικής εικόνας.

5.4 Βάση Δεδομένων

Σε αυτή την παράγραφο θα δούμε τη δομή της βάσης δεδομένων, αλλά και των ερωτημάτων μέσω των οποίων ο GeoServer μας επιστρέφει τα ίχνη που ζητάμε.

5.4.1 Δομή βάσης

Η βάση δεδομένων έχει πολύ απλή δομή. Περιέχει δύο πίνακες.

Ο GeoServer προκειμένου να μπορεί να εκτελέσει WFS insert στον πίνακα με τα ίχνη, σε βάση δεδομένων Microsoft SQL Server, χρειάζεται έναν άλλο πίνακα, που έχει μία μόνο εγγραφή, και περιγράφει ποιο είναι το μοναδικό κλειδί, και πώς και ποιος θα εισάγει τιμές για το μοναδικό κλειδί όταν εισάγονται εγγραφές. Αυτός ο πίνακας έχει μια προκαθορισμένη μορφή και μόνο μία εγγραφή που δεν αλλάζει ποτέ. Οπότε στην υπόλοιπη εργασία τον αγνοούμε.

Ο άλλος πίνακας είναι αυτός στον οποίο αποθηκεύονται όλα τα ίχνη. Τα πεδία του πίνακα αντιστοιχούν στις ιδιότητες της κλάσης Track. Με μία διαφορά. Ο Microsoft SQL Server δεν έχει τύπο δεδομένων boolean. Τις boolean μεταβλητές τις αποθηκεύει κανείς σε ένα πίνακα ως ακέραιους, με τιμή μηδέν για false και ένα για true. Ο πιο κατάλληλος τύπος δεδομένων που έχει για αυτό είναι ένας ακέραιος με μέγεθος ενός bit, όπου μπορούν να αποθηκευτούν μόνο αυτές οι δύο τιμές.

Κανονικά λοιπόν η σχεδίαση του πίνακα απαιτεί τη χρήση bit για την αποθήκευση των boolean, αλλά σε αυτή την περίπτωση υπάρχει ένα πρόβλημα με τον GeoServer. Ο GeoServer αντιστοιχεί τις τιμές bit του Microsoft SQL Server σε τιμές boolean δικές του. Έτσι, όταν για παράδειγμα ζητάμε μέσω WFS ένα ίχνος, μας επιστρέφονται τιμές true και false αντί για 0 και 1. Μέχρι εδώ όλα δουλεύουν όπως θα έπρεπε. Όταν όμως ζητάμε να εισαχθεί η τιμή true με ένα πεδίο boolean ενός ίχνους, τότε ο GeoServer, αντί να το μετατρέψει σε 1 και να το εισάγει, προσπαθεί να εισάγει το αλφαριθμητικό “true” και αποτυγχάνει. Αν πεις στον GeoServer να εισάγει το 1, τότε δεν επιχειρεί καν την εισαγωγή, γιατί περιμένει τιμές true ή false. Μέχρι να διορθωθεί το πρόβλημα σε κάποια μελλοντική έκδοση, αναγκαστικά αλλάξαμε τον τύπο των δεδομένων από bit σε short, τον αμέσως μεγαλύτερο ακέραιο τύπο δεδομένων. Αυτό φυσικά οδήγησε σε αύξηση των απαιτήσεων σε χώρο στο σκληρό δίσκο,

αλλά δεν υπήρχε άλλη επιλογή, εφόσον θέλαμε να χρησιμοποιούμε τον GeoServer με την Microsoft SQL Server μαζί.

5.4.2 Ερωτήματα

Έχουν υλοποιηθεί δύο ερωτήματα. Και τα δυο επιστρέφουν όλα τα πεδία από το μοναδικό πίνακα της βάσης δεδομένων.

Και τα δύο επιστρέφουν ένα επιπλέον πεδίο, τύπου integer, με τιμή μηδέν. Το επιπλέον πεδίο χρησιμοποιείται για να αποθηκεύσουμε την ηλικία του ίχνους που επιστρέφεται. Την τιμή του την δίνει η κλάση TrackClientMapWindowHelper όταν λαμβάνει τα ίχνη, ανάλογα με την χρονική σφραγίδα του ίχνους και το χρόνο λήψης. Κανονικά λοιπόν θα περίμενε κανείς να μη μεταδίδεται ένα άδειο πεδίο από τον GeoServer. Ο λόγος που αυτό το πεδίο μεταδίδεται (με μηδενική τιμή) εξαρχής, είναι γιατί με τα GeoTools δεν υπάρχει εύκολος και γρήγορος τρόπος να προστεθεί στη συνέχεια στον client. Θα απαιτούσε πολύ χρόνο και δεκάδες γραμμές κώδικα για μια τόσο απλή ενέργεια. Κι εφόσον ο χρόνος είναι σημαντικός παράγοντας ώστε να απεικονίζουμε όσο το δυνατόν πιο πρόσφατα ίχνη, η λύση αυτή απορίφθηκε.

Το ένα ερώτημα λοιπόν είναι όπως περιγράψαμε. Όλα τα πεδία του πίνακα, με ένα επιπλέον πεδίο για την ηλικία του ίχνους, αρχικά ίσο με μηδέν.

Το δεύτερο ερώτημα μας επιστρέφει μόνο ένα ίχνος για κάθε αεροσκάφος., το πιο πρόσφατο, αλλά τα ίδια πεδία με το προηγούμενο ερώτημα. Επίσης μας επιστρέφει ένα ακόμα επιπλέον πεδίο, το AltitudeDifference. Σε αυτό το πεδίο υπάρχει η διαφορά στο ύψος ανάμεσα στο πιο πρόσφατο και το αμέσως προηγούμενο ίχνος. Αυτή η τιμή χρειάζεται για να απεικονιστεί στο χάρτη με ένα “+” αν είναι θετική, “-” αν είναι αρνητική και ”=” αν είναι μηδέν. Αν δεν υπάρχει αμέσως προηγούμενο ίχνος η τιμή είναι μηδέν. Στην παράγραφο 6.1.3 υπάρχει αυτούσιο αυτό το ερώτημα.

Και τα δύο ερωτήματα συνδυάζονται με τα OGC φίλτρα που θα δεχθεί ο GeoServer από τον client πριν υποβληθούν στη βάση.

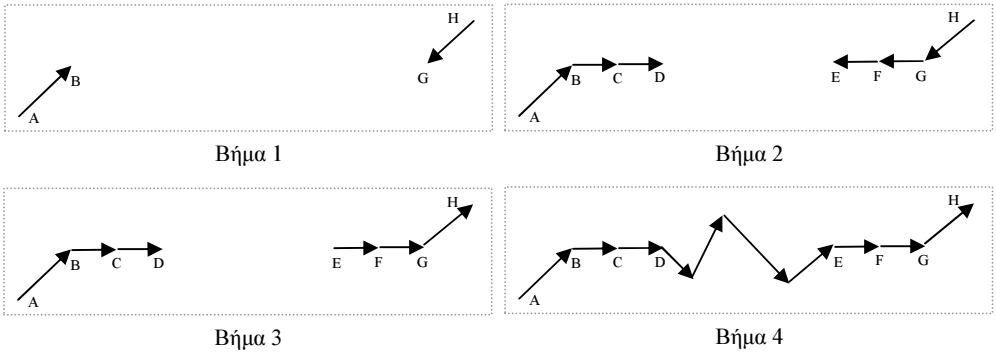
6

Υλοποίηση

6.1 Λεπτομέρειες υλοποίησης

Στις επόμενες παραγράφους θα δούμε αναλυτικότερα ορισμένα σημεία της εφαρμογής που αξίζουν την προσοχή μας.

6.1.1 Δημιουργία ενός course



Σχήμα 33: Βήματα δημιουργίας ενός course

Η κλάση CourseFactory δημιουργεί προκαθορισμένα course για να χρησιμοποιηθούν από SimulatedTargets για να ζητάει από αυτούς ο Generator τα ίχνη σε κάποια χρονική στιγμή.

Στο Σχήμα 33 βλέπουμε σχηματικά τα βήματα που ακολουθούνται για τη δημιουργία ενός course.

Τα σημεία A και H είναι αεροδρόμια, τα σημεία εκκίνησης και τερματισμού της διαδρομής. Τα βήματα είναι τα ακόλουθα:

1. Δημιουργούμε ένα CourseSegment (στο εξής «τμήμα») με τη διεύθυνση του αεροδρομίου, αρχική ταχύτητα μηδέν, με μικρή διάρκεια, με επιτάχυνση θετική και θετικό ρυθμό ανόδου. Αντιστοιχεί στην αρχική απογείωση.
2. Κάνουμε ακριβώς το ίδιο με εκκίνηση το σημείο τερματισμού. Δημιουργούμε δηλαδή μια απογείωση από το αεροδρόμιο τερματισμού. (Βήμα 1)
3. Στη συνέχεια βρίσκουμε τη διεύθυνση από το σημείο B στο σημείο G και δημιουργούμε δύο τμήματα., συνέχεια του AB, με τη νέα διεύθυνση. Το ένα έχει τόση διάρκεια ώστε να επιτύχουμε το επιθυμητό ύψος πτήσης, το άλλο έχει τόση διάρκεια ώστε να πετύχουμε τη επιθυμητή ταχύτητα. Όποιο από τα δύο έχει μικρότερη διάρκεια αποτελεί το τμήμα BC. Το άλλο απορρίπτεται.
4. Αν έχουμε φτάσει το επιθυμητό ύψος, συνεχίζουμε να επιταχύνουμε στο επόμενο τμήμα με ρυθμό ανόδου μηδέν, μέχρι να φτάσουμε την επιθυμητή ταχύτητα. Αυτό θα είναι το τμήμα CD. Αν όμως από το προηγούμενο βήμα έχουμε φτάσει την επιθυμητή ταχύτητα, συνεχίζουμε στο επόμενο τμήμα με μηδέν επιτάχυνση και τον ίδιο (σταθερό πλέον) ρυθμό ανόδου, και αυτό αποτελεί το τμήμα CD.
5. Κάνουμε το ίδιο από το σημείο G προς το B, με την αντίθετη διεύθυνση, δημιουργώντας τα τμήματα GF και FE. (Βήμα 2)
6. Αντιστρέφουμε τα τμήματα FE, GF και HG ώστε να αποτελούν την προσγείωση. (Βήμα 3)
7. Με σταθερή ταχύτητα και μηδενική επιτάχυνση και ρυθμό ανόδου, δημιουργούμε ευθύγραμμα τμήματα ανάμεσα σε όλα τα waypoints που έχουμε ορίσει ότι θα πρέπει να επισκεφτούμε σε αυτό το course. (Βήμα 4)

6.1.2 Δημιουργία WFS insert request

Παραθέτουμε αυτή τη συγκεκριμένη κλάση εδώ, καθώς υλοποιεί αρκετά σημεία που έχουμε συναντήσει ως τώρα και θα κάνει πιο εύκολη την κατανόησή τους. Επίσης είναι ένα παράδειγμα του πώς χρησιμοποιεί κανείς τα GeoTools και κατά πόσο είναι εύκολη ή όχι η χρήση τους.

Η μέθοδος run που είναι και η μοναδική της κλάσης, εκτελείται σε ένα ξεχωριστό thread για να είναι καλύτερη η απόκριση της εφαρμογής σε εξωτερικά γεγονότα (σε γεγονότα που προκαλεί ο χρήστης, όπως η μετακίνηση ενός παράθυρου).

```
private class TrackWFSSenderThread implements Runnable {

    @Override
    public void run() {
        isSending = true;
        status.setSending(true);
        GlobalLookup.getInstance().remove(status);
        GlobalLookup.getInstance().add(status);
        TrackReceiverAPI TRAPI = GlobalLookup.getInstance().getLookup().lookup(TrackReceiverAPI.class);
        if (TRAPI == null) return;
        Track currentTrack = null;
        long startTime;
        int currentBundleSize;

        while (shouldBeSending) {
            startTime = System.currentTimeMillis();
            currentBundleSize = bundleSize;
            transaction = new DefaultTransaction("trans");
            featureStore.setTransaction(transaction);
            for (int i = 0; i < currentBundleSize; i++) {
                try {
                    currentTrack = TRAPI.pollForTrack(100, TimeUnit.MILLISECONDS);
                } catch (InterruptedException ex) {}
                if (currentTrack == null) break;
                builder.set("TargetID", currentTrack.ID);
                builder.set("SAC", currentTrack.dataSource.getSAC());
                builder.set("SIC", currentTrack.dataSource.getSIC());
                builder.set("RadarDetection", currentTrack.radarDetection);
                builder.set("Simulated", currentTrack.simulated ? 1 : 0);
                builder.set("Position", geometryFactory.createPoint(new Coordinate(currentTrack.longitude, currentTrack.latitude)));
                builder.set("Speed", currentTrack.speed);
                builder.set("Heading", currentTrack.heading);
                builder.set("Altitude", currentTrack.altitude);
                builder.set("AltitudeGarbled", currentTrack.altitudeGarbled ? 1 : 0);
                builder.set("AltitudeValidated", currentTrack.altitudeValidated ? 1 : 0);
                builder.set("M2Code", currentTrack.m2Code);
                builder.set("M2Garbled", currentTrack.m2Garbled ? 1 : 0);
                builder.set("M2Smoothed", currentTrack.m2Smoothed ? 1 : 0);
                builder.set("M2Validated", currentTrack.m2Validated ? 1 : 0);
                builder.set("M3ACode", currentTrack.m3ACode);
                builder.set("M3AGarbled", currentTrack.m3AGarbled ? 1 : 0);
                builder.set("M3ASmoothed", currentTrack.m3ASmoothed ? 1 : 0);
                builder.set("M3AValidated", currentTrack.m3AValidated ? 1 : 0);
                builder.set("MillisecondsTime", currentTrack.timestamp);
                SimpleFeature newFeature = builder.buildFeature(null);
                featureCollection.add(newFeature);
            } //end for
            try {
                featureStore.addFeatures(featureCollection);
                transaction.commit();
            } catch (IOException ex) {
                printError("ERROR, Track WFS Sender sending thread: Unexpected error - " + ex.getLocalizedMessage());
                try {
                    transaction.rollback();
                } catch (IOException ex1) {
                    printError("ERROR, Track WFS Sender sending thread: Unexpected error - " + ex1.getLocalizedMessage());
                }
            } finally {
                status.setAverageTimeToSendOneTrack((System.currentTimeMillis() - startTime)/((double)currentBundleSize));
                GlobalLookup.getInstance().remove(status);
                GlobalLookup.getInstance().add(status);
                try {
                    transaction.close();
                } catch (IOException ex) {
                    printError("ERROR, Track WFS Sender sending thread: Unexpected error - " + ex.getLocalizedMessage());
                } finally {
                    featureCollection.clear();
                }
            } //end try
        } //end while
        isSending = false;
        status.setSending(false);
        GlobalLookup.getInstance().remove(status);
        GlobalLookup.getInstance().add(status);
    }
}
```

Σχήμα 34: Η κλάση TrackWFSSenderThread

Στη μέθοδο χρησιμοποιούνται ορισμένες μεταβλητές που έχουν οριστεί έξω από αυτή, ορισμένες από αυτές μάλιστα έχουν ήδη αρχικοποιηθεί έξω από αυτή.

1. isSending: boolean που τίθεται από τη διαδικασία και δείχνει την κατάστασή της, δηλαδή αν στέλνει requests ή όχι.
2. status: TrackWFSSenderStatus, μεταβλητή που περιέχει την κατάσταση του αποστολέα.
3. shouldBeSending: boolean που τίθεται εκτός της κλάσης. Αν πάρει την τιμή false, η μέθοδος run πρέπει να τερματίσει τη λειτουργία της.
4. bundleSize: volatile int, τίθεται εκτός της κλάσης και χρησιμοποιείται για να καθορίσουμε τον αριθμό των ιχνών που θα εισάγουμε σε κάθε request.
5. transaction: Transaction, κλάση των GeoTools που ορίζει ένα WFS transaction.
6. featureStore: FeatureStore<SimpleFeatureType, SimpleFeature>, κλάση των GeoTools που ορίζει μια μονάδα αποθήκευσης γεωχωρικών δεδομένων, στην περίπτωσή μας ένα layer του GeoServer, το tracks_edit_all που μας δίνει πρόσβαση ανάγνωσης/εγγραφής σε όλα τα πεδία του πίνακα την βάσης.
7. builder: SimpleFeatureBuilder, κλάση των GeoTools για το «χτίσιμο» ενός feature. Αν πούμε featureType το schema του layer του GeoServer, τότε η μεταβλητή builder αρχικοποιείται ώστε να φτιάχνει features τύπου featureType.
8. featureCollection: DefaultFeatureCollection, είναι μια κλάση των GeoTools για την αποθήκευση features στη μνήμη, και την αρχικοποιούμε ώστε να αοπθηκεύει features τύπου featureType.

Η ίδια η διαδικασία ακολουθεί τα παρακάτω βήματα:

1. Ορίζει ότι ξεκίνησε.
2. Ενημερώνει το Global lookup ότι ξεκίνησε.
3. Αναζητάει στο Global lookup κάποιο instance του TrackReceiverAPI, δηλαδή ενός δέκτη. Αν δε βρει κάποιο τερματίζει, αλλιώς συνεχίζει.
4. Στέλνει WFS insert requests όσο η μεταβλητή shouldBeSending είναι true:
    - Κρατάει το χρόνο που ξεκινάει η διαδικασία για να στείλει ένα request.
    - Αρχικοποιεί μεταβλητές
    - Ζητάει ένα ίχνος από την ουρά που έχει συνδεθεί (με το TrackReceiverAPI), χτίζει ένα feature από τα δεδομένα του με τον builder, και το προσθέτει στο featureCollection. Αυτό το βήμα το επαναλαμβάνει τόσες φορές όσο η τιμή του bundleSize.

- Προσθέτει τα features του featureCollection στο featureStore και κλείνει το transaction, οπότε και στέλνονται τα features στον GeoServer.
- Υπολογίζει το μέσο χρόνο που χρειάστηκε για να στείλει ένα feature και ενημερώνει το status και το Global lookup με τη νέα τιμή.
- Τερματίζει το transaction και αδειάζει το featureCollection, για να είναι έτοιμα για το επόμενο bundle.

5. Όταν τελικά η μεταβλητή shouldBeSending γίνει false, βγαίνει από το βρόγχο επανάληψης, ενημερώνει τις μεταβλητές isSending και status ότι τελείωσε, και αναρτά στο Global lookup την αλλαγή στην κατάστασή του.

6.1.3 Ερώτημα SQL για τα τελευταία διαθέσιμα ίχνη

Αυτό είναι από τα δύο ερωτήματα που έχουν εισαχθεί στον GeoServer προκειμένου να λαμβάνουμε ένα ίχνος από κάθε αεροσκάφος, το πιο πρόσφατο (Σχήμα 35).

Το layer που έχει εισαχθεί αυτό το ερώτημα είναι το haf:tracks_get_current.

```
SELECT
    t1.UniqueID,
    t1.TargetID,
    t1.SAC, t1.SIC,
    t1.RadarDetection,
    t1.Simulated,
    t1.Position, t1.Speed, t1.Heading,
    t1.Altitude, t1.AltitudeGarbled, t1.AltitudeValidated,
    t1.M2Code, t1.M2Garbled, t1.M2Smoothed, t1.M2Validated,
    t1.M3ACode, t1.M3AGarbled, t1.M3ASmoothed, t1.M3AValidated,
    t1.MillisecondsTime,
    0 AS MillisecondsAge,
    ISNULL((t1.Altitude - (SELECT TOP 1
                               t3.Altitude AS SecondAltitude
                           FROM
                               Tracks AS t3
                           WHERE
                               (NOT (t3.UniqueID = t1.UniqueID)) AND (t3.TargetID = t1.TargetID)
                           ORDER BY
                               t3.MillisecondsTime DESC)
          ),0) AS AltitudeDifference
FROM
    (SELECT * FROM TrackHistory.dbo.Tracks) AS t1
        JOIN
    (SELECT TargetID, MAX(MillisecondsTime) AS max_time FROM TrackHistory.dbo.Tracks GROUP BY TargetID) AS t2
        ON t1.TargetID = t2.TargetID AND t1.MillisecondsTime = t2.max_time
```

Σχήμα 35: SQL ερώτημα για τα τελευταία διαθέσιμα ίχνη

Το υποερώτημα που έχουμε ονομάσει t2 επιστρέφει ζεύγη τιμών TargetID και max_time από τον πίνακα Tracks, όπου το max_time είναι ο μεγαλύτερος χρόνος που βρέθηκε για κάθε TargetID. Αυτό το υποερώτημα δε μπορεί να επιστρέψει και τις υπόλοιπες τιμές της εγγραφής που έχει αυτό το χρόνο, καθώς δεν εφαρμόζεται πάνω τους aggregate function και δεν αποτελούν μέρος του GROUP BY. Έτσι συνδέουμε αυτά τα ζεύγη με τον πίνακα t1 (που είναι ουσιαστικά ο πίνακας Tracks) για να βρούμε τα υπόλοιπα πεδία της εγγραφής που έχει αυτό το TargetID και MillisecondsTime=max_time. Αυτό μας δίνει το τελευταίο ίχνος για κάθε διαφορετικό TargetID, δηλαδή για κάθε διαφορετικό αεροσκάφος.

Το πεδίο MillisecondsAge ορίζεται ως μηδέν. Εξηγήσαμε το λόγο στην παράγραφο 5.4.2.

Το πεδίο AltitudeDifference ορίζεται ως η διαφορά ανάμεσα στο Altitude του ίχνους που επιστρέφεται και την τιμή ενός άλλου υποερωτήματος. Το υποερώτημα αυτό επιλέγει το πεδίο Altitude από τον πίνακα Tracks. Περιορίζουμε όμως τις εγγραφές σε αυτές που έχουν το ίδιο TargetID με αυτό του ίχνους που θα επιστραφεί, από αυτές αφαιρούμε την εγγραφή που έχει το ίδιο μοναδικό κλειδί με αυτό του ίχνους που θα επιστραφεί (ώστε να μην πάρουμε το ύψος του πιο πρόσφατου ίχνους), ταξινομούμε τα αποτελέσματα ως προς τη χρονοσφραγίδα τους αντίστροφα (το πιο πρόσφατο ίχνος πρώτα) και επιλέγουμε μόνο το πρώτο αποτέλεσμα. Παίρνουμε δηλαδή το ύψος από το δεύτερο πιο πρόσφατο ίχνος για το κάθε TargetID (το πρώτο πιο πρόσφατο το έχουμε απορρίψει). Αυτό το υποερώτημα εκτελείται για κάθε ένα από τα TargetID που θα επιστραφούν, δημιουργώντας αρκετά μεγάλη επιπλέον καθυστέρηση. Αλλά το πρόσημο του AltitudeDifference πρέπει να απεικονιστεί για να έχουμε μια αεροπορική εικόνα αντίστοιχη του PALLAS. Εναλλακτικά θα μπορούσαμε να επιστρέψουμε όλα τα ίχνη μέσα στο χρονικό παράθυρο που θα ζητηθεί, και να αναλάβει ο client να κρατήσει το πιο πρόσφατο, αλλά και να υπολογίσει το AltitudeDifference. Αυτό όμως θα επέφερε μεγάλο φόρτο στην κωδικοποίηση / αποκωδικοποίηση των δεδομένων, αλλά και στη μεταφορά τους μέσω δικτύου, κάνοντας τα πράγματα πολύ χειρότερα.

6.1.4 Διαδικασία σχεδίασης ενός ίχνους

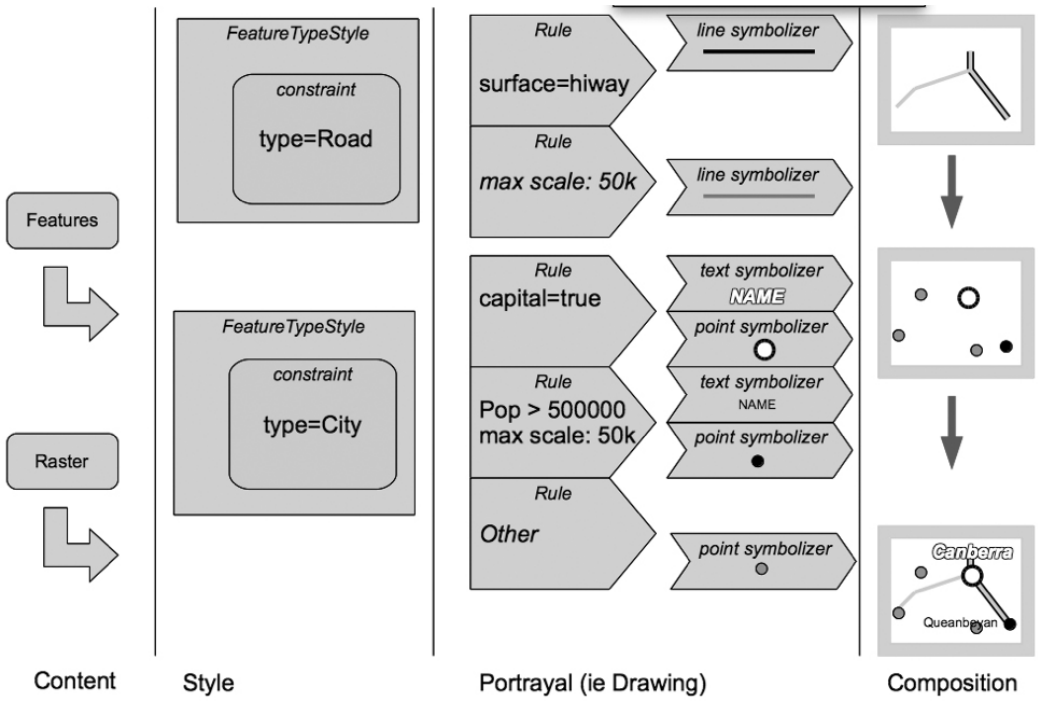
Από το (GeoTools, 2012b) δανειστήκαμε το παρακάτω σχεδιάγραμμα όπου φαίνονται τα βήματα - επίπεδα σχεδίασης ενός οποιουδήποτε feature (Σχήμα 36).

Στο πρώτο επίπεδο (αριστερά) βλέπουμε τα δεδομένα, στην περίπτωσή μας τα ίχνη.

Στο δεύτερο επίπεδο βλέπουμε τα στυλ. Αν θέλουμε τα δεδομένα μας να εμφανιστούν τελικά στην οθόνη, θα πρέπει να δημιουργήσουμε ένα στυλ σύμφωνα με το οποίο θα εμφανιστούν. Ένα στυλ δημιουργείται για ένα συγκεκριμένο τύπο δεδομένων. Εμείς έχουμε δημιουργήσει ένα στυλ για το schema των ιχνών που επιστρέφει ο GeoServer. Έχουμε επίσης δημιουργήσει ένα στυλ για κάθε shapefile που φορτώνουμε στο υπόβαθρο.

Στο τρίτο επίπεδο βλέπουμε ότι το επόμενο βήμα είναι να προσθέσουμε rules (κανόνες) σε κάθε στυλ. Δεν είναι τόσο σαφές στο σχήμα, αλλά κάθε κανόνας έχει ένα ή περισσότερα φίλτρα, και έναν ή περισσότερους symbolizers. Τα features φιλτράρονται από όλους τους κανόνες που θα βάλουμε στο στυλ. Αν ένα feature πληροί τις προδιαγραφές όλων των φίλτρων του κανόνα, τότε σχεδιάζεται σύμφωνα με τους symbolizers αυτού του κανόνα. Μπορεί λοιπόν για παράδειγμα να σχεδιαστεί από δύο symbolizers του κανόνα Α, κανέναν του Β (γιατί δεν περνάει τα φίλτρα), και έναν του κανόνα Γ. Κάθε symbolizer «σχεδιάζει» από ένα στοιχείο στην οθόνη - ένα σημείο, μια γραμμή, ένα τετράγωνο, ένα Χ, κάποιο κείμενο κτλ. Στο τέταρτο επίπεδο, συνθέτονται όσα έχουν σχεδιάσει οι symbolizers για να

δημιουργηθεί η τελική εικόνα. Για παράδειγμα, σε αυτό το στάδιο ο renderer φροντίζει τα κείμενα να μην υπερκαλύπτουν το ένα το άλλο.



Σχήμα 36: Διαδικασία σχεδίασης ενός feature

Στον κώδικα, βλέπουμε ένα πολύ μικρό μέρος αυτής της διαδικασίας στο Σχήμα 37.

```
private void addPSRSSRSymbol(FilterFactory2 ff, StyleFactory sf, String color, List<Symbolizer> symbolizers) {
    Mark markSQ = sf.getSquareMark();
    markSQ.setStroke(sf.createStroke(ff.literal(color), ff.literal(1)));
    markSQ.setFill(sf.createFill(getColorLiteral(Color.BLACK, ff), ff.literal(0.0))); //Transparent
    Graphic gr = sf.createDefaultGraphic();
    gr.graphicalSymbols().clear();
    gr.graphicalSymbols().add(markSQ);
    gr.setSize(ff.literal(11));
    symbolizers.add(sf.createPointSymbolizer(gr, null));

    Mark markX = sf.getXMark();
    markX.setStroke(sf.createStroke(ff.literal(color), ff.literal(0.0))); //Transparent
    markX.setFill(sf.createFill(ff.literal(color), ff.literal(1.0)));
    gr = sf.createDefaultGraphic();
    gr.graphicalSymbols().clear();
    gr.graphicalSymbols().add(markX);
    gr.setSize(ff.literal(11));
    symbolizers.add(sf.createPointSymbolizer(gr, null));
}
```

Σχήμα 37: Σχεδίαση μέρους ενός συμβόλου ίχνους

Αυτό το κομμάτι κώδικα, που αποτελεί μέρος της κλάσης TrackClientMapWindowHelper, προσθέτει δύο symbolizers στη λίστα των symbolizers που συνθέτουν την εικόνα ενός ίχνους. Ο ένας σχεδιάζει ένα τετράγωνο, και ο άλλος ένα Χ. Έτσι συμβολίζονται τα αεροσκάφη στο

PALLAS όταν υπάρχουν πληροφορίες για αυτά από primary και secondary radar ταυτόχρονα. Η λίστα σε άλλο σημείο εμπλουτίζεται με τρεις ακόμα symbolizers, έναν για κάθε γραμμή κειμένου που συνοδεύει το ίχνος (TargetID, ύψος και ταχύτητα). Σύνολο 5 symbolizers για κάθε ίχνος. Για την ακρίβεια, υπάρχει 1 στυλ, με 52 κανόνες, που ο καθένας έχει 1 φίλτρο που ελέγχει την ηλικία του ίχνους και τι είδους κεραίες το ανίχνευσαν, και 5 symbolizers που σχεδιάζουν το ίχνος. Δημιουργούνται δηλαδή συνολικά 260 symbolizers και το κάθε ίχνος ελέγχεται αν περνάει 104 κανόνες.

6.2 Εγκατάσταση

Σημαντικό είναι να αναφέρουμε το εξής. Για την καλύτερη απόδοση των υποσυστημάτων, θα μπορούσε το κάθε ένα να εκτελείται σε διαφορετικό μηχάνημα, αν απαιτεί ιδιαίτερη επεξεργαστική ισχύ ή μνήμη. Κατά την ανάπτυξη και δοκιμή της εφαρμογής όμως δεν ήταν διαθέσιμο ένα ανάλογο πλήθος υπολογιστών. Έτσι όλα τα modules αναπτύχθηκαν σε μία και μοναδική εφαρμογή. Επίσης, δεν υπήρχε νόημα δημιουργίας εκτελέσιμων αρχείων ή αρχείων εγκατάστασης, από τη στιγμή που ανάλογα με το περιβάλλον που θα τοποθετηθεί το σύστημα, ανάλογα θα χωριστεί η εφαρμογή σε πολλά ή λίγα τμήματα. Όσον αφορά λοιπόν την εγκατάσταση της εφαρμογής, θα αναφερθούμε γενικά στο πώς θα μπορούσε να γίνει σε διάφορες περιπτώσεις.

Στα βήματα της εγκατάστασης που ακολουθούν, όποτε αναφέρεται η εγκατάσταση ενός προγράμματος, το πρόγραμμα αυτό θα βρίσκεται στο συνοδευτικό CD σε φάκελο ίδιο με το όνομα του προγράμματος. Τα βήματα λοιπόν είναι ως εξής:

1. Εγκαθιστούμε σε ένα μηχάνημα το JDK.
2. Εγκαθιστούμε στο ίδιο μηχάνημα το NetBeans IDE.
3. Μεταφέρουμε τον φάκελο TrackProjectAllModules από το CD σε ένα φάκελο του ίδιου μηχανήματος, και βγάζουμε το read-only attribute από όλα τα αρχεία.
4. Ανοίγουμε το NetBeans IDE και ανοίγουμε το project TrackProjectAllModules από τον φάκελο όπου το τοποθετήσαμε.
5. Ελέγχουμε τη διαδρομή για τις βιβλιοθήκες (GeoTools και JCalendar) ώστε να βεβαιωθούμε ότι είναι σωστή. Αν δεν είναι τη διορθώνουμε.
6. Εφόσον έχουμε αποφασίσει σε ποια μηχανήματα θα τοποθετηθεί κάθε τμήμα της εφαρμογής που θα δημιουργήσουμε, εισάγουμε τις κατάλληλες IP στο Settings.java
7. Ξεκινάμε να δημιουργούμε ένα ένα τα τμήματα της εφαρμογής που θα χρησιμοποιήσουμε.

- Από το project αφαιρούμε τα modules που δε θα μας χρειαστούν. Αν για παράδειγμα θέλουμε να φτιάξουμε την εφαρμογή client, τότε κρατάμε μόνο τα modules που φαίνονται στο Σχήμα 10.
- Κάνουμε δεξί κλικ στο project, Package as, ZIP Distribution
- Όταν τελειώσει η δημιουργία της εφαρμογής, από τον φάκελο TrackProjectAllModules/dist, αντιγράφουμε το αρχείο zip στον υπολογιστή όπου θα το χρησιμοποιήσουμε.
- Φροντίζουμε σε εκείνον τον υπολογιστή να υπάρχει το τελευταίο JRE ή JDK.
- Αποσυμπιέζουμε τον φάκελο trackprojectallmodules από το αρχείο zip.

8. Επαναλαμβάνουμε το βήμα 7 για όλες τις εφαρμογές που θα δημιουργήσουμε.
9. Στο μηχάνημα που έχουμε αποφασίσει ότι θα μπει η βάση δεδομένων, εγκαθιστούμε τον Microsoft SQL Server 2008 R2 Express.
10. Ανοίγουμε το SQL Server Management Studio και δημιουργούμε μια βάση δεδομένων με το όνομα TrackHistory.
11. Τρέχουμε το script από το φάκελο script του CD, μέσα από το SQL Server Management Studio. Αυτό θα δημιουργήσει τους πίνακες και του λογαριασμούς χρηστών που απαιτούνται.
12. Στο ίδιο μηχάνημα εγκαθιστούμε τον GeoServer.
13. Αντιγράφουμε το φάκελο haf από το CD στο φάκελο ... \GeoServer 2.2\data_dir\workspaces\haf. Αυτό θα δημιουργήσει τα απαραίτητα layers στον GeoServer.
14. Ξεκινάμε στον GeoServer.
15. Τρέχουμε τις εφαρμογές μας με double-click στο αρχείο ..\trackprojectallmodules\bin\ trackprojectallmodules.exe

7

Έλεγχος

Σε αυτό το κεφάλαιο γίνεται ο έλεγχος της εφαρμογής, όπου και βλέπουμε πως αποδίδει κάτω από διάφορες συνθήκες.

7.1 Μεθοδολογία ελέγχου

Ο πλήρης έλεγχος μιας εφαρμογής σαν αυτή χρειάζεται πάρα πολύ χρόνο και εγκαταστάσεις. Καθώς υπάρχουν πολλοί συνδυασμοί με τους οποίους μπορείς να κατανέμεις την εφαρμογή σε διάφορους υπολογιστές, και κάθε φορά η απόδοσή της διαφοροποιείται (ειδικά αν μεταβάλλουμε και την απόσταση των υπολογιστών), ο έλεγχος για να γίνει σωστά πρέπει να γίνει από beta testers σε διαφορετικά και ανομοιογενή συστήματα.

Σε αυτή την εργασία, καθώς δεν υπήρχε αυτή η ευχέρεια, αποφασίστηκε ο έλεγχος να γίνει σε ένα μηχάνημα, όπου όλα θα είναι τοποθετημένα εκεί: ο Microsoft SQL Server, ο GeoServer, το JDK, το NetBeans IDE, η εφαρμογή μας (με όλα τα modules εγκατεστημένα).

Το μηχάνημα είναι ένας μετρίων δυνατοτήτων προσωπικός υπολογιστής.

Κατά τον έλεγχο θα χρησιμοποιούμε το παραθυρικό περιβάλλον για να ενεργοποιούμε και να ρυθμίζουμε ένα ένα τα υποσυστήματα, και θα παρατηρούμε τη συμπεριφορά τους κάτω από διαφορετικές ρυθμίσεις.

7.2 Αναλυτική παρουσίαση ελέγχου

7.2.1 Γεννήτρια ιχνών

Για να ελέγξουμε τη γεννήτρια, ανοίγουμε τα παράθυρα Generator settings και Generator status από το μενού της εφαρμογής. Στη συνέχεια ακολουθούμε τα παρακάτω βήματα.

1. Ρυθμίζουμε τον αριθμό των ιχνών στο ελάχιστο και το μεσοδιάστημα που θα μεσολαβεί ανάμεσα σε δύο διαδοχικές παραγωγές στο μέγιστο, χρησιμοποιώντας τα ρυθμιστικά του παράθυρου Generator settings.
2. Ενεργοποιούμε τη γεννήτρια από το toggle button του παράθυρου Generator settings.
3. Στο παράθυρο Generator status βλέπουμε αν έχει ενεργοποιηθεί η γεννήτρια, πόσο είναι το πραγματικό μεσοδιάστημα και πόσα ίχνη παράγονται σε κάθε «γύρο» παραγωγής. Παρατηρούμε ότι πράγματι έχει ενεργοποιηθεί, και ότι οι τιμές για το μεσοδιάστημα και τα ίχνη είναι σχεδόν ίδιες με τις ρυθμίσεις μας.
4. Σταδιακά αυξάνουμε τον αριθμό των ιχνών και μειώνουμε το μεσοδιάστημα και βλέπουμε αν η γεννήτρια ανταποκρίνεται ή δε μπορεί να παράγει αρκετά γρήγορα ίχνη. Στο μηχάνημα που έγινε ο έλεγχος, μπορούσε να παράγει 500 ίχνη ανά 10ms χωρίς να χρησιμοποιείται πάνω από το 5% της επεξεργαστικής ισχύος του υπολογιστή.
5. Σταδιακά μειώνουμε τον αριθμό των ιχνών από τις ρυθμίσεις. Μετά από λίγα λεπτά παρατηρούμε να μειώνεται και στο status ο αριθμός τους, καθώς ένα ένα τα αεροσκάφη προσγειώνονται.
6. Κλείνουμε τη γεννήτρια, αλλάζουμε τις ρυθμίσεις, και την ξανανοίγουμε. Παρατηρούμε ότι δουλεύει κανονικά.

Με τον παραπάνω έλεγχο διαπιστώνουμε ότι η γεννήτρια φαίνεται να δουλεύει σωστά. Στην πραγματικότητα θα μπορούσε να μη δουλεύει ο κώδικας και να μην παράγει τίποτα, δίνοντάς μας λαθεμένη εντύπωση. Οπότε ο έλεγχος της γεννήτριας θα είναι ολοκληρωμένος όταν θα δούμε πραγματικά (αργότερα στη διαδικασία ελέγχου) τα ίχνη να εισάγονται στη βάση δεδομένων.

7.2.2 Δέκτης Μονάδας Εισόδου

Για να ελέγξουμε το δέκτη, ανοίγουμε το παράθυρο Track receiver window από το μενού της εφαρμογής.

1. Φροντίζουμε να είναι κλειστή η γεννήτρια ιχνών.

2. Πατάμε το πλήκτρο Start receiving για να ενεργοποιήσουμε το δέκτη. Βλέπουμε στο παράθυρο Generator status ότι έχει συνδεθεί με τη γεννήτρια, αλλά δε λαμβάνει ίχνη, που είναι φυσιολογικό αφού δεν παράγονται ίχνη.
3. Ανοίγουμε τη γεννήτρια ιχνών και παρατηρούμε ότι ο αριθμός των ιχνών που συγκεντρώνονται στο buffer του δέκτη μεγαλώνει συνεχώς.
4. Σταματάμε τη γεννήτρια και βλέπουμε το buffer να διατηρεί σταθερό το πλήθος από ίχνη που συγκρατεί.
5. Κλείνουμε και ανοίγουμε τον δέκτη μερικές φορές συνεχόμενα. Παρατηρούμε ότι ο στο παράθυρο Generator status δείχνει να είναι συνδεδεμένοι πολλοί δέκτες. Είναι μια εικονική ένδειξη. Αν ξεκινήσουμε τη γεννήτρια, βλέπουμε ότι προσπαθεί να στείλει ίχνη σε όλους τους συνδεδεμένους δέκτες και τότε αντιλαμβάνεται από όλοι εκτός από έναν είναι αποσυνδεμένοι και ενημερώνει το status ανάλογα. Η λάθος ένδειξη δημιουργείται γιατί δεν κάνει κανέναν έλεγχο για να δει αν συνεχίζει κάποιος δέκτης να είναι συνδεδεμένος, μέχρι να χρειαστεί τελικά να του στείλει δεδομένα.
6. Αφήνουμε τη γεννήτρια και το δέκτη ανοιχτό λίγη ώρα και παρατηρούμε ότι το buffer δε μεγαλώνει πάνω από 1000. Η τιμή αυτή ορίζεται από το Settings.java προκειμένου να μην καταναλώνεται άσκοπα η μνήμη του υπολογιστή.

Διαπιστώνουμε λοιπόν ότι ο δέκτης δουλεύει όπως ακριβώς θα έπρεπε. Από το μέγεθος του buffer που αυξάνεται έχουμε και μια πρώτη ένδειξη ότι πράγματι η γεννήτρια παράγει ίχνη.

7.2.3 Αποστολέας Μονάδας Εισόδου

Για να ελέγξουμε τον αποστολέα, ανοίγουμε τα παράθυρα Track WFS sender settings και Track WFS sender status.

1. Ξεκινάμε τη γεννήτρια και τον δέκτη και αφήνουμε να μαζευτούν αρκετά ίχνη στο buffer του δέκτη.
2. Κλείνουμε τη γεννήτρια και τον δέκτη.
3. Πατάμε το πλήκτρο Start sending tracks to WFS server, έχοντας ρυθμίσει το Bundle size στη μικρότερη δυνατή τιμή. Παρατηρούμε σιγά σιγά το buffer να αδειάζει καθώς τα ίχνη στέλνονται στον server.
4. Ανοίγουμε τη γεννήτρια με το μικρότερο δυνατό ρυθμό παραγωγής, και στη συνέχεια ανοίγουμε και το δέκτη. Παρατηρούμε ότι το buffer του δέκτη παραμένει (σχεδόν συνέχεια) μηδενικό.
5. Αλλάζουμε τις ρυθμίσεις για το μεσοδιάστημα παραγωγής ιχνών, τον αριθμό των ιχνών και το Bundle size και παρατηρούμε ότι παραδόξως, ένα bundle size 40 ή 50

μας δίνει τα καλύτερα αποτελέσματα, που στον υπολογιστή της δοκιμής φτάνει τα 130 ίχνη ανά δευτερόλεπτο.

7.2.4 Μονάδα Αποθήκευσης

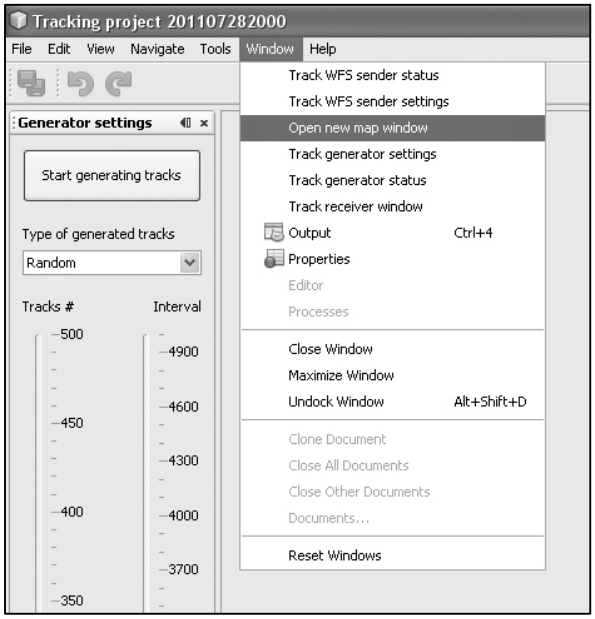
Για τον έλεγχο της Μονάδας Αποθήκευσης, ακολουθούμε τα παρακάτω βήματα:

1. Ανοίγουμε το SQL Server Management Studio. Εκτελούμε ένα ερώτημα διαγραφής όλων των δεδομένων του πίνακα ιχνών.
2. Ενεργοποιούμε τη γεννήτρια, το δέκτη και τον αποστολέα για ένα μεγάλο χρονικό διάστημα (πχ 30 λεπτά).
3. Κλείνουμε τη γεννήτρια, το δέκτη και τον αποστολέα.
4. Από το SQL Server Management Studio εκτελούμε ερωτήματα πάνω στα δεδομένα του πίνακα ιχνών και ελέγχουμε τα αποτελέσματα. Αν για παράδειγμα είχαμε ρυθμίσει τη γεννήτρια με μικρό μεσοδιάστημα, τότε τα ίχνη για ένα συγκεκριμένο TrackID θα πρέπει να απέχουν (χρονικά) μεταξύ τους κατά περίπου αυτό το μεσοδιάστημα, θα πρέπει να υπάρχουν αρκετά ίχνη που θα έχουν συνεχώς αυξανόμενο ύψος, τα περισσότερα θα πρέπει να έχουν σταθερό ύψος, και τα πιο πρόσφατα θα πρέπει να έχουν συνεχώς μειούμενο ύψος. Μπορούμε να εκτελέσουμε πλήθος ελέγχων.
5. Αν διαθέτουμε το ArcGIS της ESRI ή άλλη εφαρμογή που να διαβάσει τις πληροφορίες του πίνακα ιχνών με τη χωρική τους διάσταση, τότε μπορούμε να προβάλουμε τα περιεχόμενα του πίνακα στο χώρο και να δούμε κατά πόσο ανταποκρίνονται σε όσα περιγράψαμε στην παράγραφο 6.1.1.

7.2.5 Μονάδα Παρουσίασης

Για τον έλεγχο της Μονάδας Παρουσίασης εκτελούμε τα ακόλουθα βήματα:

1. Ξεκινάμε τη γεννήτρια, τον δέκτη και τον αποστολέα, σε χαμηλούς ρυθμούς παραγωγής ιχνών.
2. Επιλέγουμε στο μενού της εφαρμογής την εντολή

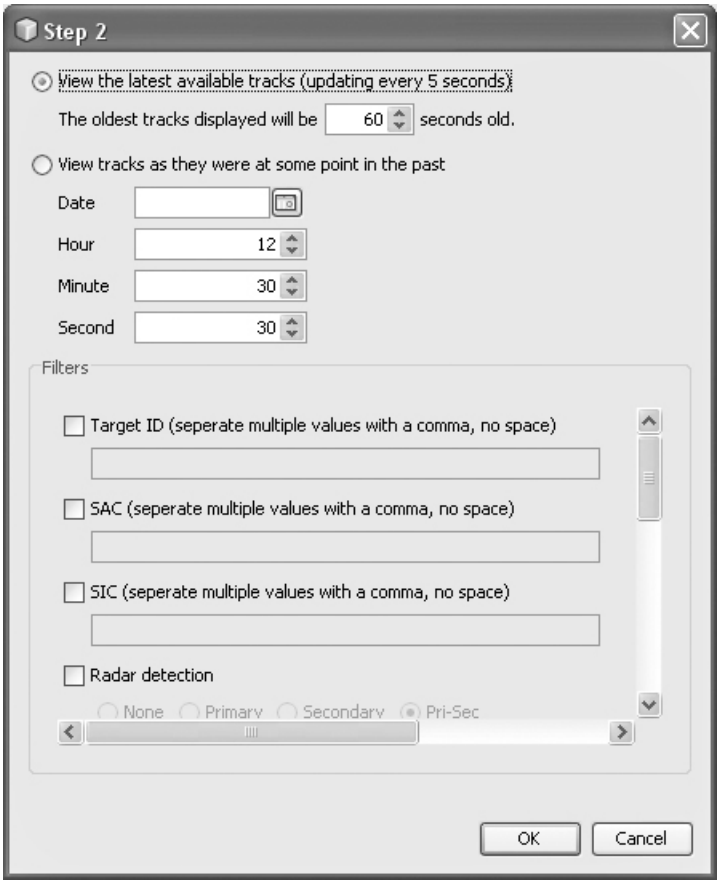


Σχήμα 38: Επιλογές μενού Window

Window -> Open new map window (Σχήμα 38). Εμφανίζεται ένα παράθυρο διαλόγου όπως στο Σχήμα 39.

Σχήμα 39: Διάλογος για την επιλογή είδους δέκτη

3. Στο παράθυρο διαλόγου που εμφανίστηκε, μόνο η πρώτη επιλογή είναι ενεργή. Όπως έχουμε πει, η δεύτερη επιλογή υπάρχει ως επίδειξη της ευκολίας χρήσης εναλλακτικών δεκτών. Θα μπορούσε να είναι ένας δέκτης που τα ίχνη τα μετατρέπει σε multilines προκειμένου να παρουσιάσει τις πορείες των αεροσκαφών, ή απλά να λαμβάνει ίχνη από διαφορετικό server. Με αυτό το παράθυρο διαλόγου θα επιλέγουμε τον επιθυμητό δέκτη κάθε φορά. Επιλέγουμε λοιπόν View tracks και πατάμε OK.



Σχήμα 40: Διάλογος για την εισαγωγή φίλτρων

4. Στο νέο παράθυρο διαλόγου που εμφανίζεται (Σχήμα 40) δεν αλλάζουμε καμία από τις επιλογές και πατάμε OK.
5. Εμφανίζεται το παράθυρο με τον χάρτη. Περιμένουμε μερικά δευτερόλεπτα για την αρχικοποίηση και τη λήψη - σχεδίαση των πρώτων ιχνών (Σχήμα 41).

Σχήμα 41: Πλήρης άποψη της εφαρμογής σε λειτουργία

6. Χρησιμοποιούμε τα εργαλεία της μπάρας εργαλείων στο παράθυρο του χάρτη για να κάνουμε zoom in, zoom out, pan, να δούμε πληροφορίες για τα layers σε κάποιο σημείο, ή (με το τελευταίο εργαλείο) να δούμε την πλήρη έκταση του χάρτη.
7. Παρατηρούμε τα ίχνη να ανανεώνονται κάθε 5 δευτερόλεπτα, οπότε και σχεδιάζονται στις νέες τους θέσεις.
8. Περιμένουμε μερικά λεπτά μέχρι να προσγειωθεί κάποιο από τα απεικονιζόμενα αεροσκάφη, οπότε και βλέπουμε σταδιακά το χρώμα απεικόνισης να σκουραίνει μέχρι το ένα λεπτό αργότερα, οπότε και το ίχνος εξαφανίζεται.
9. Κλείνουμε το παράθυρο του χάρτη και επαναλαμβάνουμε τα βήματα 2-8, επιλέγοντας όμως στο βήμα 3 διαφορετικές κάθε φορά παραμέτρους, ώστε να ελέγξουμε τη λειτουργία των φίλτρων.
10. Ανοίγουμε ταυτόχρονα και δεύτερο παράθυρο χάρτη και παρατηρούμε ότι τα ίχνη και στους δύο χάρτες σχεδιάζονται με το σωστό τρόπο.
11. Κλείνουμε τα παράθυρα του χάρτη και επαναλαμβάνουμε τα βήματα 1-10, επιλέγοντας όμως στο βήμα 1 σταδιακά μεγαλύτερους ρυθμούς παραγωγής ιχνών. Παρατηρούμε ότι καθώς αυξάνεται ο φόρτος της Μονάδας Αποθήκευσης, αδυνατεί να ανταποκριθεί αρκετά γρήγορα στα αιτήματα εισαγωγής και ανάκτησης δεδομένων.

8

Επίλογος

Στα προηγούμενα κεφάλαια είδαμε το στόχο μας, τον αναλύσαμε, σχεδιάσαμε τη λύση, την υλοποιήσαμε και την ελέγξαμε. Σε αυτό το κεφάλαιο θα δούμε συνοπτικά τη γενική εικόνα της εφαρμογής, και πώς αυτή θα μπορούσε στο μέλλον να βελτιωθεί.

8.1 Σύνοψη και συμπεράσματα

Ο στόχος που θέσαμε από την αρχή ήταν «η δημιουργία μιας εφαρμογής που θα μπορεί να λαμβάνει πληροφορίες ιχνών αεροσκαφών από διάφορες πηγές, θα τις αποθηκεύει σε μια γεωχωρική βάση δεδομένων, και θα τις προβάλει στον τελικό χρήστη». Ας δούμε τα θετικά και τα αρνητικά σημεία της υλοποίησής μας.

8.1.1 Θετικά σημεία

1. Η εφαρμογή μας μπορεί να λαμβάνει πληροφορίες ιχνών αεροσκαφών από πολλές πηγές ταυτόχρονα.
2. Μπορεί να τις αποθηκεύει σε μια γεωχωρική βάση δεδομένων.
3. Μπορεί να τις προβάλλει στον τελικό χρήστη.
4. Είναι εύκολα επεκτάσιμη.
5. Έχει μεγάλη ευελιξία ως προς τα ποια τμήματά της θα δουλεύουν στον ίδιο υπολογιστή.

6. Έχει αναπτυχθεί σε ένα σύγχρονο παραθυρικό περιβάλλον.
7. Η διαδικασία αναβάθμισης είναι εύκολη και μπορεί να γίνει χωρίς επανεγκατάσταση της εφαρμογής.
8. Χρησιμοποιεί δωρεάν λογισμικό σε όλα τα επίπεδα (εκτός από το λειτουργικό σύστημα).

8.1.2 Αρνητικά σημεία

1. Ο συνδυασμός WFS - Geoserver - Microsoft SQL Server είναι πολύ αργός. Η εφαρμογή θα πρέπει να εγκατασταθεί σε πραγματικά γρήγορους server προκειμένου να ανταπεξέλθει σε πραγματικές συνθήκες. Δεν είναι όμως τόσο αργός ώστε να είναι απαγορευτικός.
2. Το Παράθυρο Παρουσίασης είναι πολύ φτωχό. Θα μπορούσε να οπτικοποιεί πολύ περισσότερες πληροφορίες, να εμφανίζει ολόκληρες τροχιές αεροσκαφών (ζωντανά ή στο παρελθόν), να έχει πιο ποιοτικά δεδομένα για το υπόβαθρο.
3. Η έκδοση της βάσης δεδομένων που χρησιμοποιήσαμε μπορεί να ήταν δωρεάν αλλά έχει ένα περιορισμό που εν μέρει μας απασχολεί. Έχει όριο αποθήκευσης τα 10GB. Με τους ρυθμούς αποθήκευσης που θα είχαμε σε πραγματική λειτουργία, το όριο αυτό θα ήταν θέμα ημερών να το φτάσουμε. Αν όμως η εφαρμογή χρησιμοποιηθεί με πραγματικά δεδομένα, θα χρησιμοποιηθεί και η πλήρης έκδοση του Microsoft SQL Server, οπότε και δε θα υπάρχει αυτό το πρόβλημα. Θα υπάρχει πρόβλημα όμως στην τήρηση ιστορικού για μεγάλο χρονικό διάστημα, καθώς θα φτάνουμε τα όρια αποθήκευσης του σκληρού δίσκου. Και δεν έχει προβλεφθεί διαδικασία μεταφοράς του ελέγχου αποθήκευσης σε κάποιο άλλο μηχανισμό μέχρι να γίνει εγκατάσταση νέου αποθηκευτικού μέσου έτσι ώστε να μη χαθούν πληροφορίες.

8.1.3 Συμπεράσματα

Η εφαρμογή υπερκάλυψε το στόχο που θέσαμε. Απέχει πολύ όμως από το να αντικαταστήσει το υπάρχον σύστημα παρακολούθησης ιχνών, το PALLAS. Μπορεί όμως να χρησιμοποιείται παράλληλα. Αν μάλιστα δε χρησιμοποιείται για ζωντανή προβολή της αεροπορικής εικόνας, τότε μειώνεται κατακόρυφα ο φόρτος της Μονάδας Αποθήκευσης. Αυτό την κάνει ιδανική ως ένα σύστημα τήρησης ιστορικού με δυνατότητα προβολής του. Η ευκολία προσθήκης μελλοντικών επεκτάσεων την κάνει ακόμα πιο ελκυστική, και ικανή να καλύψει οποιαδήποτε απαίτηση προκύψει.

8.2 Μελλοντικές επεκτάσεις

Η μοναδική ευελιξία της εφαρμογής στην προσθήκη επεκτάσεων θα μπορούσε να βρει εφαρμογή σε κάθε τμήμα της. Ας δούμε ενδεικτικά μερικές ιδέες.

1. Η Μονάδα Εισόδου μπορεί να επεκταθεί προσθέτοντας επιπλέον Ανεξάρτητες Μονάδες Εισόδου ώστε να μπορεί να λαμβάνει ίχνη από το πρωτόκολλο του PALLAS και του ASTERIX.
2. Όλες οι Ανεξάρτητες Μονάδες Εισόδου μπορούν να επεκταθούν ώστε να υποστηρίζουν και άλλα πρωτόκολλα εκτός από το WFS, ώστε να επιτυγχάνουμε καλύτερες ταχύτητες μεταφοράς και αποθήκευσης δεδομένων.
3. Μπορεί να προστεθεί ένα ενδιάμεσο επίπεδο ανάμεσα στη Μονάδα Εισόδου και τη Μονάδα Αποθήκευσης ώστε να μοιράζει τα ίχνη σε πολλές Μονάδες Αποθήκευσης (ως backup ή για διαμοιρασμό του φόρτου).
4. Η Μονάδα Αποθήκευσης μπορεί να επεκταθεί ώστε να υποστηρίζει και άλλα πρωτόκολλα εκτός από το WFS.
5. Η Μονάδα Αποθήκευσης μπορεί να επεκταθεί ώστε να υποστηρίζει αποθήκευση σε περισσότερες βάσεις δεδομένων ταυτόχρονα (ως backup ή για διαμοιρασμό του φόρτου).
6. Η Μονάδα Αποθήκευσης μπορεί να επεκταθεί ώστε να υποστηρίζει ομαλή μετάβαση από ένα μέσο αποθήκευσης σε ένα άλλο.
7. Η Μονάδα Παρουσίασης μπορεί να επεκταθεί ώστε να προσφέρει περισσότερες επιλογές στον τρόπο εμφάνισης των αποτελεσμάτων και του υπόβαθρου. Θα μπορούσε για παράδειγμα να έχει χρωματικά θέματα να διαλέξουμε, ή δυνατότητα τρισδιάστατης προβολής.
8. Η Μονάδα Παρουσίασης μπορεί να επεκταθεί ώστε να υποστηρίζει και άλλα πρωτόκολλα εκτός του WFS.
9. Η Μονάδα Παρουσίασης μπορεί να επεκταθεί ώστε να μπορεί να συνδεθεί με περισσότερους από έναν server ταυτόχρονα, και να προβάλλει ίχνη από πολλούς server σε ένα χάρτη ταυτόχρονα.
10. Η Μονάδα Παρουσίασης μπορεί να επεκταθεί ώστε να μπορεί να προβάλλει (προαιρετικά) την απόσταση κάθε ίχνους από το κοντινότερο σε αυτό ίχνος, και να μας ειδοποιεί όταν αυτή μικραίνει πολύ.
11. Ολόκληρη η εφαρμογή μπορεί να επεκταθεί με την ανάπτυξη επιπλέον Μονάδων Παρουσίασης όπου κάθε μία θα καλύπτει διαφορετικές ανάγκες (πχ Web browser based).

12. Ολόκληρη η εφαρμογή μπορεί να επεκταθεί ώστε να μπορεί να διαχειρίζεται και ίχνη οχημάτων από GPS εκτός από ίχνη αεροσκαφών.
13. Ολόκληρη η εφαρμογή μπορεί να επεκταθεί ώστε να υλοποιεί όλες τις λειτουργίες του PALLAS και να το αντικαταστήσει.
14. Ολόκληρη η εφαρμογή μπορεί να επεκταθεί ώστε να τηρεί στατιστικά στοιχεία (αποστάσεις που διανύθηκαν, ώρες πτήσης που εκτελέστηκαν, καύσιμα που χρησιμοποιήθηκαν κ.α.) και να τα προβάλλει στον τελικό χρήστη.

9

Βιβλιογραφία

1. Δρακόπουλος Γεώργιος, & Παναγιώτου Γεώργιος (2002). *Παροχή Αεροπορικής Εικόνας PALLAS στην ΠΑ*. Αθήνα: Υπηρεσία Αεροπορικών Εκδόσεων
2. Malensek, M., Pallickara, S. L., & Pallickara, S. (2012). *Exploiting geospatial and chronological characteristics in data streams to enable efficient storage and retrievals*. Future Generation Computer Systems, Available online 12 June 2012, ISSN 0167-739X, 10.1016/j.future.2012.05.024. [accessed 18/10/2012]

   http://www.sciencedirect.com/science/article/pii/S0167739X1200132X
3. TomTom Business Solutions. *Fleet management solutions.* [online][accessed 18/10/2012] http://business.tomtom.com/en_gb/fleet-management
4. iLink (2010). *PowerFleet fleet management system.* Online [accessed 18/10/2012] http://www.powerfleet.gr
5. Patroumpas, K., & Sellis, T. (2004, August). Managing trajectories of moving objects as data streams. In *Spatio-Temporal Database Management, 2nd International Workshop STDBM'04, Toronto, Canada, August* (Vol. 30).
6. Microsoft (2012a). *Microsoft SQL Server Previous Editions.* Online [accessed 18/10/2012]

   http://www.microsoft.com/sqlserver/en/us/editions/previous-versions.aspx
7. OGC (2012a). *Web Feature Service.* Online [accessed 18/10/2012]

   http://www.opengeospatial.org/standards/wfs
8. OGC (2012b). *About OGC.* Online [accessed 18/10/2012]

http://www.opengeospatial.org/ogc

9. bowens, Holmes C., Shorter C., Aime A., Pumphrey M., Williq I. & Paxil (2009, December). *What is GeoServer.* Online [accessed 18/10/2012] http://geoserver.org/display/GEOS/What+is+Geoserver
10. OSGeo. *The Open Source Geospatial Foundation….* Online [accessed 18/10/2012] http://www.osgeo.org
11. GeoTools (2012a). *About GeoTools.* Online [accessed 18/10/2012] http://www.geotools.org/about.html
12. Oracle (2012a). *Java SE Downloads.* Online [accessed 18/10/2012] http://www.oracle.com/technetwork/java/javase/downloads/index.html
13. Oracle (2012, June). *The Java Tutorials.* Online [accessed 18/10/2012] http://docs.oracle.com/javase/tutorial
14. Oracle (2012b). *NetBeans IDE - The Smarter and Faster Way to Code.* Online [accessed 18/10/2012] http://netbeans.org/features/index.html
15. Oracle (2012c). *The NetBeans Platform.* Online [accessed 18/10/2012] http://netbeans.org/features/platform/index.html
16. Microsoft (2012b). *Microsoft JDBC Driver for SQL Server.* Online [accessed 18/10/2012] http://msdn.microsoft.com/en-us/sqlserver/aa937724.aspx
18. DIVA-GIS. *DIVA-GIS*. Online [accessed 18/10/2012] http://www.diva-gis.org
19. Kai Tödter (2012). *JCalendar.* Online [accessed 18/10/2012] http://www.toedter.com/en/jcalendar
20. EUROCONTROL (2012a). *ASTERIX - What is ASTERIX?* Online [accessed 18/10/2012] http://www.eurocontrol.int/services/asterix
21. EUROCONTROL (2012b). *Specifications - Documents, Latest issues of ASTERIX Documents.* Online [accessed 18/10/2012] http://www.eurocontrol.int/services/specifications-documents
22. GeoTools (2012b). *Map Style Tutorial.* Online [accessed 18/10/2012] http://docs.geotools.org/latest/userguide/tutorial/map/style.html