Υλοποίηση Συστήματος Θαλάσσιας Παρακολούθησης

SPEAKER NOTES

Ξεκινάω slides

1. Καλησπέρα σας. Ονομάζομαι Γιώργος Κορύλλος και θα σας παρουσιάσω σήμερα το πώς εγώ και η ομάδα μου σχεδιάζουμε να υλοποιήσουμε το Σύστημα Θαλάσσιας Παρακολούθησης.

   Θα εστιάσουμε στη σύνδεση των απαιτήσεων, όπως ορίστηκαν στο έγγραφο SRS, με τη δομή που αποτυπώνεται στο UML διάγραμμα κλάσεων και την αρχιτεκτονική που επιλέξαμε.
   [Επόμενο slide]

2. Το σύστημά μας, όπως περιγράφεται στο SRS, πρέπει να καλύψει αρκετές βασικές λειτουργίες: Να δείχνει πλοία ζωντανά στον χάρτη παίρνοντας δεδομένα AIS, να διαχειρίζεται χρήστες με διαφορετικούς ρόλους, να επιτρέπει την οργάνωση πλοίων σε στόλους και το φιλτράρισμα, να παρακολουθεί ζώνες για παραβιάσεις κανόνων και να στέλνει ειδοποιήσεις, συν τη δυνατότητα ανίχνευσης συγκρούσεων.

   Αυτές οι λειτουργίες φέρνουν συγκεκριμένες τεχνικές προκλήσεις: Πώς διαχειριζόμαστε την τεράστια και συνεχόμενη ροή δεδομένων AIS; Πώς κρατάμε τον χάρτη στο frontend

γρήγορο και συγχρονισμένο; Πώς γίνεται αποδοτικά η επεξεργασία των κανόνων και ο εντοπισμός συμβάντων στο backend; Πώς εξασφαλίζουμε ότι το σύστημα μπορεί να μεγαλώσει αν χρειαστεί και πώς προστατεύουμε τα δεδομένα και διαχειριζόμαστε τα δικαιώματα των χρηστών;
[Επόμενο slide]

3. Για να απαντήσουμε στις προκλήσεις, υιοθετούμε μια διπλή προσέγγιση. Πρώτον, ορίζουμε καθαρά τη δομή του συστήματος με αυτό το UML Class Diagram. Αυτό μας δίνει ένα μοντέλο για τις βασικές έννοιες – τα πλοία, τους χρήστες, τις ζώνες – και πώς συνδέονται. Είναι η κοινή μας γλώσσα.

   Δεύτερον, επιλέγουμε μια ευέλικτη αρχιτεκτονική microservices, που τρέχουν σε Docker. Αυτό φαίνεται σχηματικά εδώ. Χρησιμοποιούμε React με Leaflet για το frontend, Spring Boot για το backend API, Kafka για τη διαχείριση της ροής δεδομένων AIS, PostgreSQL για τη βάση, και WebSockets για την επικοινωνία σε πραγματικό χρόνο.

   Στις επόμενες διαφάνειες, θα δείξουμε ακριβώς πώς αυτό το μοντέλο (UML) και αυτή η αρχιτεκτονική συνεργάζονται για να υλοποιήσουν τις λειτουργίες που είδαμε πριν.
   [Επόμενο slide]

4. Για τα δεδομένα AIS και την απεικόνισή τους: Η ροή ξεκινά από το Kafka. Ένα microservice στο backend 'ακούει' για νέα μηνύματα AIS, τα επεξεργάζεται και τα αντιστοιχίζει στις UML κλάσεις μας: AISData για τη θέση/ταχύτητα, συνδεδεμένο με το

Ship **[Next για εικόνα 2 φορές]**. Το Ship έχει και τα στατικά του δεδομένα (ShipStaticData).

Αυτό το backend service μετά 'σπρώχνει' τις ενημερώσεις θέσης μέσω WebSockets στο frontend. Το React λαμβάνει αυτά τα μηνύματα και, χρησιμοποιώντας το Leaflet, ενημερώνει άμεσα τη θέση και την κατεύθυνση των πλοίων στον χάρτη.

Όταν ο χρήστης κάνει κλικ σε ένα πλοίο, το frontend κάνει ένα REST call στο backend για να πάρει περισσότερες λεπτομέρειες (όνομα, τύπος κλπ) **[Next για εικόνα]** ή το ιστορικό πορείας, το οποίο θα σχεδιάζεται ως γραμμή στον χάρτη.
[Επόμενο slide]

5. Για τους χρήστες: Χρησιμοποιούμε Spring Security στο backend για έλεγχο ταυτότητας και εξουσιοδότηση, με τα στοιχεία να αποθηκεύονται με ασφάλεια (hashed passwords) σε PostgreSQL.

   Οι UML κλάσεις User, RegisteredUser, Admin και το enum UserRole μοντελοποιούν τους διαφορετικούς τύπους χρηστών και τα δικαιώματά τους **[Next για εικόνα 2 φορές]**. Το frontend παρέχει τις φόρμες εισόδου και εγγραφής που βλέπετε στα wireframes.

   Οι εγγεγραμμένοι χρήστες μπορούν να δημιουργήσουν στόλους. Στο UML, η κλάση Fleet συνδέει τον χρήστη με τα πλοία του **[Next για εικόνα]**. Το backend προσφέρει REST endpoints για τη δημιουργία, ανάκτηση, ενημέρωση και διαγραφή αυτών των στόλων.

Για το φιλτράρισμα, η κλάση Filter κρατά τα κριτήρια **[Next για εικόνα]**. Το backend εφαρμόζει αυτά τα φίλτρα στα δεδομένα πριν τα στείλει, ώστε ο χρήστης να βλέπει μόνο ό,τι ζήτησε.

Τέλος **[Next για εικόνα]**, οι διαχειριστές έχουν πρόσβαση σε ένα προστατευμένο REST endpoint για να τροποποιούν στατικά δεδομένα πλοίων, όπως τον τύπο τους.
[Επόμενο slide]

6. Οι χρήστες ορίζουν Ζώνες Ενδιαφέροντος σχεδιάζοντας ένα πολύγωνο στον χάρτη μέσω του Leaflet στο frontend. Θέτουν επίσης κανόνες, όπως μέγιστη ταχύτητα. Αυτά αποθηκεύονται στο backend μέσω REST.

   [Next για εικόνα] Στο UML, η ZoneOfInterest περιέχει το Polygon (μια λίστα γεωγραφικών σημείων) και μια λίστα από Restriction (που το καθένα έχει τύπο και τιμή).

   Η ανίχνευση γίνεται στο backend: Ένα service ελέγχει συνεχώς ποια πλοία βρίσκονται μέσα σε ποιες ενεργές ζώνες, χρησιμοποιώντας γεωχωρικές συναρτήσεις. Αν ένα πλοίο είναι μέσα, ελέγχει αν παραβιάζει κάποιον από τους περιορισμούς της ζώνης, π.χ., αν η ταχύτητά του από το τελευταίο AISData είναι μεγαλύτερη από το όριο.

   Αν βρεθεί παραβίαση, δημιουργούμε ένα αντικείμενο Violation, όπως ορίζεται στο UML, που καταγράφει ποιο πλοίο, σε ποια ζώνη, ποιον κανόνα παραβίασε και πότε. Η δημιουργία αυτού του αντικειμένου πυροδοτεί τη διαδικασία ειδοποίησης.
   [Επόμενο slide]

7. Οι ειδοποιήσεις ενεργοποιούνται όταν δημιουργηθεί ένα αντικείμενο Violation ή CollisionRisk. Το backend τότε δημιουργεί ένα αντικείμενο Notification, σύμφωνα με το UML μοντέλο μας, που περιέχει πληροφορίες για το συμβάν.

   [Next για εικόνα] Αυτή η ειδοποίηση στέλνεται μέσω WebSockets στον κατάλληλο χρήστη και εμφανίζεται στο frontend του, όπως βλέπουμε στο wireframe.

   Για την ανίχνευση συγκρούσεων (bonus λειτουργία): Ο χρήστης ορίζει μια ειδική ζώνη. Ένα backend service αναγνωρίζει τα πλοία μέσα στη ζώνη και, χρησιμοποιώντας τα δεδομένα κίνησής τους από τα AISData, προσπαθεί να προβλέψει τις μελλοντικές τους πορείες. Αν προβλέψει ότι δύο ή περισσότερα πλοία θα πλησιάσουν επικίνδυνα (κάτω από ένα όριο απόστασης σε ένα χρονικό ορίζοντα), δημιουργεί ένα αντικείμενο CollisionRisk.

   [Next για εικόνα] Στο UML, το CollisionRisk συνδέει τα εμπλεκόμενα πλοία, τη ζώνη και τον εκτιμώμενο χρόνο σύγκρουσης. Και πάλι, η δημιουργία του οδηγεί σε ειδοποίηση. **[Επόμενο slide]**

8. Αυτή είναι η βασική δομή των κλάσεων του συστήματος θαλάσσιας παρακολούθησης, η οποία αποτυπώνει τις κύριες οντότητες και τις μεταξύ τους σχέσεις.

   Ξεκινώντας από την κλάση User, που αντιπροσωπεύει τους χρήστες του συστήματος, έχουμε τρεις τύπους χρηστών: Anonymous, RegisteredUser και Admin, οι οποίοι ορίζονται μέσω

του enum UserRole. Οι εγγεγραμμένοι χρήστες μπορούν να δημιουργούν και να διαχειρίζονται προσωπικούς στόλους πλοίων και να ορίζουν περιοχές ενδιαφέροντος, ενώ οι διαχειριστές έχουν επιπλέον δικαιώματα, όπως η τροποποίηση των στατικών στοιχείων των πλοίων μέσω της μεθόδου modifyShipStaticData().

Η κλάση Ship περιλαμβάνει τόσο τα στατικά στοιχεία, όπως το μήκος, το πλάτος και τη σημαία, που αποθηκεύονται στην κλάση ShipStaticData, όσο και τα δυναμικά δεδομένα AIS, που καταγράφονται στην κλάση AISData. Κάθε πλοίο έχει πολλαπλές εγγραφές AIS που περιγράφουν τη θέση, την ταχύτητα και την πορεία του σε διαφορετικές χρονικές στιγμές.

Οι χρήστες μπορούν να οργανώνουν πλοία σε Fleet, δηλαδή προσωπικούς στόλους, και να ορίζουν ZoneOfInterest, δηλαδή περιοχές ενδιαφέροντος στον χάρτη, οι οποίες περιλαμβάνουν γεωμετρικά δεδομένα (πολύγωνα) και περιορισμούς, όπως μέγιστη επιτρεπόμενη ταχύτητα. Οι περιορισμοί αυτοί ορίζονται μέσω της κλάσης Restriction και του enum RestrictionType.

Το σύστημα παρακολουθεί παραβιάσεις αυτών των περιορισμών, που καταγράφονται στην κλάση Violation, και δημιουργεί ειδοποιήσεις (Notification) για τους χρήστες, οι οποίες μπορεί να αφορούν παραβιάσεις ή κινδύνους σύγκρουσης.

Για την παρακολούθηση συγκρούσεων, υπάρχουν οι κλάσεις CollisionMonitoringZone και CollisionRisk, όπου ορίζονται ζώνες παρακολούθησης και καταγράφονται πιθανοί κίνδυνοι σύγκρουσης μεταξύ πλοίων.

Τέλος, οι χρήστες μπορούν να εφαρμόζουν φίλτρα (Filter) για να περιορίσουν την προβολή πλοίων στον χάρτη, βάσει τύπου πλοίου ή προσωπικού στόλου.

Συνολικά, το διάγραμμα αυτό αντικατοπτρίζει με σαφήνεια τις βασικές οντότητες και τις λειτουργίες του συστήματος, υποστηρίζοντας την παρακολούθηση, τη διαχείριση και την ειδοποίηση σχετικά με τη ναυτιλιακή δραστηριότητα σε πραγματικό χρόνο.
[Επόμενο slide]

9. Το UML ενσωματώνει όλες τις βασικές έννοιες που αναλύσαμε: τους χρήστες, τα πλοία με τα στατικά και δυναμικά τους δεδομένα, τις γεωγραφικές ζώνες και τους κανόνες τους, τα συμβάντα όπως οι παραβιάσεις και οι κίνδυνοι σύγκρουσης, τις ειδοποιήσεις, καθώς και τις δομές οργάνωσης όπως οι στόλοι και τα φίλτρα.

   Τα οφέλη αυτού του μοντέλου είναι σημαντικά: Μας δίνει μια ξεκάθαρη εικόνα της δομής των δεδομένων, εξασφαλίζει ότι όλη η ομάδα έχει μια κοινή κατανόηση, καθοδηγεί άμεσα τη δημιουργία των Java κλάσεων στο backend μας, και αποτελεί τη βάση για τον σχεδιασμό των REST APIs και των μηνυμάτων που θα ανταλλάσσονται μέσω WebSockets.
   [Επόμενο slide]

10. Η αρχιτεκτονική που υλοποιούμε βασίζεται σε microservices, πακεταρισμένα με Docker και ενορχηστρωμένα με Docker Compose. Αυτό μας επιτρέπει να έχουμε ξεχωριστές, αυτόνομες

υπηρεσίες για κάθε κύρια λειτουργία: μία για τους χρήστες και την ασφάλεια, μία για την επεξεργασία των εισερχόμενων δεδομένων AIS από το Kafka, μία για την κύρια επιχειρησιακή λογική και το REST API, και μία για τη διαχείριση των WebSockets συνδέσεων.

Οι τεχνολογίες που χρησιμοποιούμε είναι σύγχρονες και δοκιμασμένες: React με Leaflet στο frontend, Spring Boot με Java στο backend, Kafka για τη ροή δεδομένων, WebSockets για την άμεση επικοινωνία, PostgreSQL για τη βάση δεδομένων χρηστών και Docker για την τυποποίηση του περιβάλλοντος.

Αυτή η αρχιτεκτονική προσφέρει σημαντικά πλεονεκτήματα: Μπορούμε να κλιμακώσουμε κάθε υπηρεσία ανεξάρτητα ανάλογα με το φορτίο, υποστηρίζει την απόδοση σε πραγματικό χρόνο που χρειαζόμαστε, και κάνει το σύστημα πιο εύκολο στη συντήρηση και την εξέλιξη.
[Επόμενο slide]

11. Πέρα από τις λειτουργίες, καλύπτουμε και τις μη λειτουργικές απαιτήσεις. Για την απόδοση, η ασύγχρονη επεξεργασία μέσω Kafka και η αποδοτική μετάδοση δεδομένων με WebSockets είναι κλειδιά, μαζί με βελτιστοποιήσεις στη βάση δεδομένων.

    Για την ασφάλεια, χρησιμοποιούμε HTTPS για την επικοινωνία, το Spring Security για έλεγχο ταυτότητας και εξουσιοδότηση, κρυπτογραφούμε τους κωδικούς (hashing) και προστατεύουμε τα API endpoints βάσει ρόλων.

Η χρηστικότητα εξασφαλίζεται με τη χρήση React για ένα μοντέρνο και responsive περιβάλλον, ακολουθώντας τον σχεδιασμό που είδαμε στα wireframes, και υποστηρίζοντας τις απαιτούμενες γλώσσες.

Τέλος, η συμβατότητα με τους σύγχρονους φυλλομετρητές είναι δεδομένη με τη χρήση του React.
[Επόμενο slide]

12. Συνοψίζοντας, παρουσιάσαμε πώς το UML διάγραμμα κλάσεων και η επιλεγμένη αρχιτεκτονική microservices μας δίνουν ένα σαφές και υλοποιήσιμο σχέδιο για να καλύψουμε τις απαιτήσεις του SRS. Συνδυάζουμε τον δομημένο σχεδιασμό που μας προσφέρει το UML με σύγχρονες τεχνολογίες όπως Kafka, React και Spring Boot.

    Αυτή η προσέγγιση μας επιτρέπει να στοχεύουμε στη δημιουργία ενός συστήματος που δεν είναι μόνο λειτουργικό, αλλά και αποδοτικό, κλιμακούμενο και ασφαλές. Τα επόμενα βήματά μας είναι η σταδιακή υλοποίηση των services, η ανάπτυξη του frontend και φυσικά οι απαραίτητοι έλεγχοι.
    [Επόμενο slide]

13. Σας ευχαριστώ πολύ για την προσοχή σας.
    [Επόμενο slide]

14. Είμαι στη διάθεσή σας για οποιεσδήποτε ερωτήσεις.
    [Τέλος]