HY359: Διαδικτυοκεντρικός Προγραμματισμός

Project Report

Team Members

Χρήστος Πολυματίδης csd5209

Ιωάννης Χατζηαντωνίου csd5193

Περιεχόμενα

Διαδικτυακό πληροφοριακό σύστημα έκτακτης ανάγκης E-199

1. Εισαγωγή

1.1 Σκοπός της αναφοράς

Η παρούσα αναφορά περιγράφει τη διαδικασία ανάπτυξης του **E-199**, ενός διαδικτυακού πληροφοριακού συστήματος που επιτρέπει τη **διαχείριση και αναφορά περιστατικών έκτακτης ανάγκης**. Το σύστημα στοχεύει στην **καλύτερη οργάνωση και ανταπόκριση των αρχών**, όπως η **πυροσβεστική υπηρεσία** και οι **εθελοντές διασώστες**, καθώς και στην ενημέρωση των πολιτών.

1.2 Περιγραφή του project

Το **E-199** επιτρέπει στους χρήστες να **καταχωρούν περιστατικά (π.χ. πυρκαγιές, τροχαία ατυχήματα)** μέσω μιας **διαδικτυακής πλατφόρμας**, ενώ η πληροφορία εμφανίζεται σε έναν **διαδραστικό χάρτη**. Οι **αρμόδιες αρχές** μπορούν να εγκρίνουν ή να απορρίψουν τις αναφορές, να **αναθέσουν εργασίες σε εθελοντές**, και να ενημερώνουν τους πολίτες σε πραγματικό χρόνο.

2. Σχεδιαστική Προσέγγιση

2.1 Γενική περιγραφή της αρχιτεκτονικής

Το **E-199** ακολουθεί την **αρχιτεκτονική client-server**, όπου το **frontend** είναι υλοποιημένο με **HTML, CSS, και JavaScript (Leaflet.js για χαρτογράφηση)**, ενώ το **backend** βασίζεται σε **Java Servlets και MySQL** για την αποθήκευση των περιστατικών. Η επικοινωνία γίνεται μέσω **REST API** και **AJAX requests**.

Επιπλέον, για τη **διαχείριση των HTTP αιτημάτων** και τη βελτίωση της απόδοσης του backend, χρησιμοποιούμε το **Spark Java Framework**. Το **Spark** επιτρέπει τη δημιουργία **lightweight REST APIs** με λιγότερο boilerplate κώδικα, διευκολύνοντας την ανάπτυξη και τη διαχείριση των endpoints. Συνδυάζεται με το **Gson** για την εύκολη μετατροπή των αντικειμένων της Java σε JSON και αντίστροφα.

2.2 Επιλογή τεχνολογιών

- **Backend:** Java Servlets, Spark Java, MySQL, Gson για μετατροπή JSON
- **Frontend:** HTML, CSS (Bootstrap), JavaScript (Leaflet.js)
- **Επικοινωνία:** AJAX, REST API μέσω Spark και Servlets
- **Χαρτογράφηση:** OpenStreetMap μέσω Leaflet.js
- **Βελτίωση επιδόσεων:** Spark Java για lightweight RESTful API ανάπτυξη

3. Σχεδίαση Back-end

3.1 Περιγραφή κλάσεων και μεθόδων

3.1.1 Κλάση `Incident`

Αντιπροσωπεύει ένα **περιστατικό** που περιλαμβάνει δεδομένα όπως τύπος συμβάντος, περιγραφή, τοποθεσία, και κατάσταση, κλπ.

Ιδιότητα	Περιγραφή
`incident_id`	Αναγνωριστικό συμβάντος
`incident_type`	Τύπος συμβάντος
`description`	Περιγραφή του συμβάντος
`user_phone`	Τηλέφωνο χρήστη που το κατέγραψε
`user_type`	Τύπος χρήστη
`address`, `prefecture`, `municipality`	Τοποθεσία του συμβάντος
`start_datetime`, `end_datetime`	Χρονικές πληροφορίες έναρξης και λήξης
`danger`	Επίπεδο επικινδυνότητας
`status`	Κατάσταση του συμβάντος
`finalResult`	Τελικό αποτέλεσμα του περιστατικού
`lat`, `lon`	Συντεταγμένες GPS
`vehicles`, `firemen`	Στοιχεία οχημάτων και πυροσβεστών

Μέθοδοι:

- **Getters & Setters** για όλες τις παραπάνω ιδιότητες.
- `setStart_datetime()`, `setEnd_datetime()`: Ορίζουν αυτόματα την ημερομηνία σε `yyyy/MM/dd HH:mm:ss`.

3.1.2 Κλάση `Message`

Αντιπροσωπεύει ένα **μήνυμα επικοινωνίας** μεταξύ χρηστών σχετικά με ένα περιστατικό.

Ιδιότητα	Περιγραφή
`message_id`	Αναγνωριστικό μηνύματος
`incident_id`	Αναφορά σε περιστατικό
`message`	Το περιεχόμενο του μηνύματος
`date_time`	Χρόνος αποστολής του μηνύματος
`sender`	Αποστολέας του μηνύματος
`recipient`	Παραλήπτης του μηνύματος

Μέθοδοι:

- **Getters & Setters** για όλες τις ιδιότητες.
- `setDate_time()`: Ορίζει αυτόματα την τρέχουσα ημερομηνία και ώρα.

3.1.3 Κλάση `Participant`

Αντιπροσωπεύει έναν **συμμετέχοντα** σε ένα περιστατικό αλλά το χρησιμοποιούμαι και για να αναπαραστήσουμε τα requests που κάνουν οι volunteers για να λάβουν μέρος σε ένα περιστατικό.

Ιδιότητα	Περιγραφή
`participant_id`	Αναγνωριστικό συμμετέχοντα
`incident_id`	Αναφορά στο σχετικό περιστατικό
`volunteer_type`	Τύπος εθελοντή
`volunteer_username`	Όνομα χρήστη εθελοντή
`status`	Κατάσταση συμμετοχής
`success`	Εάν η συμμετοχή ήταν επιτυχής
`comment`	Σχόλιο σχετικά με την παρέμβαση

Μέθοδοι:

- **Getters & Setters** για όλες τις ιδιότητες.

3.1.4 Κλάση `User`

Αντιπροσωπεύει έναν **απλό χρήστη** του συστήματος.

Ιδιότητα	Περιγραφή
`user_id`	Αναγνωριστικό χρήστη
`username`	Όνομα χρήστη
`email`, `password`	Στοιχεία πρόσβασης
`firstname`, `lastname`, `birthdate`	Προσωπικές πληροφορίες
`gender`, `job`, `afm`	Φύλο, επάγγελμα και ΑΦΜ
`country`, `address`	Διεύθυνση και χώρα διαμονής
`lat`, `lon`	Γεωγραφικές συντεταγμένες
`telephone`	Τηλέφωνο επικοινωνίας
`municipality`, `prefecture`	Τοποθεσία διαμονής

Μέθοδοι:

- **Getters & Setters** για όλες τις ιδιότητες.

3.1.5 Κλάση `Volunteer`

Αντιπροσωπεύει έναν **εθελοντή πυροσβέστη**, που είναι ειδικός τύπος χρήστη.

Ιδιότητα	Περιγραφή
`volunteer_id`	Αναγνωριστικό εθελοντή
`username`, `email`, `password`	Στοιχεία πρόσβασης
`firstname`, `lastname`, `birthdate`	Προσωπικές πληροφορίες
`gender`, `job`, `afm`	Φύλο, επάγγελμα και ΑΦΜ
`country`, `address`	Διεύθυνση και χώρα διαμονής
`lat`, `lon`, `height`, `weight`	Γεωγραφικές και φυσικές πληροφορίες
`telephone`	Τηλέφωνο επικοινωνίας
`municipality`, `prefecture`	Τοποθεσία διαμονής
`volunteer_type`	Τύπος εθελοντή

Μέθοδοι:

- **Getters & Setters** για όλες τις ιδιότητες.

3.1.6 Edit Tables – CRUD Operations

Ακολουθούν περιγραφές βασικών μεθόδων στις **EditTables**.

3.1.6.1 EditIncidentsTable

- **addIncidentFromJSON(String json):** Δημιουργεί νέο περιστατικό από JSON.
- **updateIncident(String id, HashMap updates):** Ενημερώνει ένα περιστατικό.
- **deleteIncident(String id):** Διαγράφει ένα περιστατικό από τη βάση.
- **databaseToIncidents():** Παίρνει όλα τα περιστατικά από τη βάση.
- **databaseToIncident(int IncidentId):** Επιστρέφει ένα συγκεκριμένο περιστατικό από τη βάση δεδομένων με βάση το `incident_id`.
- **incidentExists(String id):** Επιστρέφει `true` αν υπάρχει περιστατικό με το δοθέν `incident_id`, αλλιώς `false`.

3.1.6.2 EditMessagesTable

- **addMessageFromJSON(String json):** Προσθέτει νέο μήνυμα από JSON.
- **databaseToMessage(int incident_id):** Παίρνει όλα τα μηνύματα σχετιζόμενα με ένα περιστατικό.
- **getMessagesByUser(String username, int incident_id):** Επιστρέφει όλα τα μηνύματα στα οποία εμπλέκεται ο χρήστης είτε ως αποστολέας είτε ως παραλήπτης.
- **databaseForVolunteers(int incident_id, String username):** Επιστρέφει μηνύματα που μπορούν να δουν οι εθελοντές, με βάση το `incident_id`.

3.1.6.3 EditParticipantsTable

- **addParticipantFromJSON(String json):** Δημιουργεί νέο συμμετέχοντα.
- **acceptParticipant(int participantID, String username):** Αποδέχεται έναν συμμετέχοντα.
- **finalStatusParticipant(int participantID, String success, String comment):** Οριστικοποιεί το αποτέλεσμα ενός συμμετέχοντα.
- **databaseToParticipants():** Επιστρέφει όλους τους συμμετέχοντες που έχουν κατάσταση "requested".

- **databaseToAcceptedParticipants():** Επιστρέφει όλους τους συμμετέχοντες που έχουν αποδεχθεί να συμμετάσχουν σε περιστατικά.
- **databaseToRequestedParticipants():** Επιστρέφει τους συμμετέχοντες που έχουν κάνει αίτημα (`requested`).
- **updateParticipantStatus(int participantId, String status):** Ενημερώνει την κατάσταση ενός συμμετέχοντα.
- **updateParticipantComment(int participantId, String comment):** Ενημερώνει το σχόλιο ενός συμμετέχοντα.
- **updateParticipantsStatus(int incidentId, String newStatus):** Ενημερώνει την κατάσταση όλων των συμμετεχόντων για ένα συγκεκριμένο περιστατικό.

3.1.6.4 EditUsersTable

- **addUserFromJSON(String json):** Δημιουργεί νέο χρήστη από JSON.
- **updateUser(String username, String key, String value):** Ενημερώνει στοιχεία χρήστη.
- **databaseToUsers(String username, String password):** Παίρνει στοιχεία χρήστη από τη βάση.

3.1.6.5 EditVolunteersTable

- **addVolunteerFromJSON(String json):** Προσθέτει νέο εθελοντή από JSON.
- **updateVolunteer(String username, String personalpage):** Ενημερώνει πληροφορίες εθελοντή.
- **databaseToVolunteer(String username, String password):** Παίρνει τα δεδομένα ενός εθελοντή.

3.2 Servlets

3.2.1 Servlets του admin

1. LoginAdmin

Σκοπός:
Το `LoginAdmin` servlet είναι υπεύθυνο για τη διαδικασία σύνδεσης του διαχειριστή (admin) στο σύστημα.

Κύριες Λειτουργίες:

- **Μέθοδος GET:**
    - Λαμβάνει το **password** ως παράμετρο από το αίτημα.
    - Το όνομα χρήστη (username) είναι προκαθορισμένο ως "**admin**".
    - Συνδέεται στη βάση δεδομένων και ελέγχει αν υπάρχει εγγραφή με το συγκεκριμένο username και password στον πίνακα `users`.
    - Αν η αυθεντικοποίηση είναι επιτυχής, επιστρέφει ένα JSON αντικείμενο που περιέχει τα στοιχεία του admin.
    - Αν αποτύχει, επιστρέφει μια απάντηση που δηλώνει ότι η σύνδεση δεν ήταν επιτυχής.
- **Μέθοδος POST:**
    - Λαμβάνει ένα JSON request που περιέχει **username και νέο password**.
    - Εκτελεί ένα SQL update στον πίνακα `users` για να αλλάξει τον κωδικό πρόσβασης του admin.
    - Αν η αλλαγή είναι επιτυχής, επιστρέφει μήνυμα επιτυχίας και ενημερώνει το cookie του admin.
    - Αν αποτύχει, επιστρέφει μήνυμα αποτυχίας.

---

2. AdminParticipantsServlet

Σκοπός:
Το `AdminParticipantsServlet` επιτρέπει στον διαχειριστή να διαχειρίζεται τις αιτήσεις συμμετοχής των εθελοντών σε περιστατικά.

Κύριες Λειτουργίες:

- **Μέθοδος GET:**

    - Λαμβάνει ως παράμετρο το `status`, το οποίο μπορεί να είναι `"requested"` ή `"accepted"`.
    - Αν το `status` είναι `"accepted"`, επιστρέφει όλους τους αποδεκτούς συμμετέχοντες.
    - Αν το `status` είναι `"requested"`, επιστρέφει όλους τους εθελοντές που έχουν υποβάλει αίτηση αλλά δεν έχουν ακόμα εγκριθεί.
    - Αν η παράμετρος `status` λείπει ή είναι μη έγκυρη, επιστρέφει μήνυμα λάθους.
- **Μέθοδος PUT:**
    - Λαμβάνει ένα JSON αντικείμενο που περιέχει **τον κωδικό του συμμετέχοντα και την κατάσταση αποδοχής**.
    - Αν ο συμμετέχοντας γίνει `"accepted"`, ενημερώνει το περιστατικό μειώνοντας τον αριθμό των απαιτούμενων **πυροσβεστών ή οδηγών**.
    - Αν γίνει `"rejected"`, ενημερώνει τη βάση δεδομένων ώστε ο συμμετέχοντας να απορριφθεί.
    - Αν η εγγραφή του συμμετέχοντα δεν βρεθεί, επιστρέφει μήνυμα λάθους.
- **Μέθοδος POST:**
    - Λαμβάνει ένα JSON αντικείμενο που περιέχει ένα περιστατικό (`incident`).
    - Αν το περιστατικό έχει τεθεί σε κατάσταση `"finished"`, ενημερώνει όλους τους συμμετέχοντες σε αυτό ώστε το status τους να γίνει `"finished"`.
    - Αν η αίτηση είναι λανθασμένη, επιστρέφει μήνυμα λάθους.

---

3. StatisticsServlet

Σκοπός:
Το `StatisticsServlet` παρέχει στατιστικά δεδομένα σχετικά με τα περιστατικά και τους εθελοντές στο σύστημα.

Κύριες Λειτουργίες:

- **Μέθοδος GET:**
    - Εκτελεί ερωτήματα SQL για να συλλέξει δεδομένα σχετικά με:
        - **Αριθμό περιστατικών ανά τύπο (π.χ., φωτιά, τροχαίο, πλημμύρα).**
        - **Αριθμό εθελοντών ανά ρόλο (π.χ., οδηγοί, διασώστες).**

        - **Συνολική χρήση πόρων, όπως οχήματα και πυροσβέστες.**
    - Επιστρέφει ένα JSON αντικείμενο που περιλαμβάνει αυτά τα στατιστικά.
    - Αν προκύψει σφάλμα στη βάση δεδομένων, επιστρέφει μήνυμα λάθους.

3.2.2 Servlets του Volunteer

1. CookieVolunteerServlet

Σκοπός:
Το `CookieVolunteerServlet` είναι υπεύθυνο για τη διαχείριση των συνεδριών χρηστών μέσω cookies HTTP. Διασφαλίζει ότι οι εθελοντές παραμένουν συνδεδεμένοι κατά τη διάρκεια των αλληλεπιδράσεών τους με το σύστημα.

Κύριες Λειτουργίες:

- **Μέθοδος GET:**
    - Ανακτά το cookie `volunteerData`, εάν υπάρχει, για να επαληθεύσει μια ενεργή συνεδρία.
    - Επιστρέφει μια απάντηση JSON που υποδεικνύει εάν ο χρήστης είναι συνδεδεμένος.
- **Μέθοδος POST:**
    - Διαγράφει το cookie `volunteerData`, πραγματοποιώντας αποσύνδεση του χρήστη.
    - Εξασφαλίζει σωστή διαχείριση συνεδρίας για την αποτροπή μη εξουσιοδοτημένης πρόσβασης.

2. HistoryIncidentsServlet

Σκοπός:
Αυτό το servlet παρέχει ένα ιστορικό καταγεγραμμένων περιστατικών στα οποία έχει συμμετάσχει ένας εθελοντής, επιτρέποντάς του να παρακολουθεί τη δραστηριότητά του και τη συνεισφορά του.

Κύριες Λειτουργίες:

- **Μέθοδος GET:**
    - Ανακτά όλα τα παρελθοντικά περιστατικά στα οποία ο εθελοντής έχει χαρακτηριστεί ως "αποδεκτός" στο σύστημα.
    - Πραγματοποιεί ερωτήματα στη βάση δεδομένων για την ανάκτηση λεπτομερειών περιστατικών βάσει του ιστορικού συμμετοχής.
    - Επιστρέφει μια απάντηση JSON που περιέχει όλες τις σχετικές πληροφορίες για τα περιστατικά.

---

3. LoginVolunteerServlet

Σκοπός:
Το `LoginVolunteerServlet` διαχειρίζεται τη διαδικασία σύνδεσης και ενημέρωσης προφίλ των εθελοντών, εξασφαλίζοντας ασφαλή πρόσβαση στο σύστημα.

Κύριες Λειτουργίες:

- **Μέθοδος GET:**
  - Επαληθεύει τα διαπιστευτήρια των χρηστών συγκρίνοντάς τα με τα αποθηκευμένα στη βάση δεδομένων στον πίνακα `volunteers`.
  - Εάν η επαλήθευση είναι επιτυχής, αποθηκεύει βασικές πληροφορίες του εθελοντή σε ένα JSON-encoded cookie (`volunteerData`).
- **Μέθοδος POST:**
  - Ενημερώνει τα προσωπικά δεδομένα του εθελοντή, όπως όνομα, email, διεύθυνση και στοιχεία επικοινωνίας.

---

4. CheckParticipantsServlet

Σκοπός:
Αυτό το servlet ελέγχει τις αιτήσεις συμμετοχής των εθελοντών σε συγκεκριμένα περιστατικά.

Κύριες Λειτουργίες:

- **Μέθοδος GET:**
  - Αποδέχεται τα `volunteer_username` και `incident_id` ως παραμέτρους.
  - Ανακτά όλες τις αιτήσεις συμμετοχής από τη βάση δεδομένων και φιλτράρει όσες αντιστοιχούν στον συγκεκριμένο εθελοντή και περιστατικό με κατάσταση "ζητήθηκε (requested)".
  - Επιστρέφει μια απάντηση JSON με τις σχετικές συμμετοχές.

---

5. ParticipantsServlet

Σκοπός:
Το `ParticipantsServlet` διαχειρίζεται τις αιτήσεις συμμετοχής των εθελοντών σε περιστατικά.

Κύριες Λειτουργίες:

- **Μέθοδος GET:**
  - Ανακτά όλες τις αιτήσεις συμμετοχής από τη βάση δεδομένων και τις επιστρέφει σε JSON μορφή.
- **Μέθοδος POST:**
  - Δέχεται μια νέα αίτηση συμμετοχής σε μορφή JSON.

- Καταχωρεί την αίτηση στη βάση δεδομένων για περαιτέρω επεξεργασία.

---

6. RunningIncidentsServlet

Σκοπός:
Αυτό το servlet παρέχει μια λίστα των τρεχόντων περιστατικών στα οποία μπορούν να συμμετάσχουν οι εθελοντές, βάσει του ρόλου τους (π.χ., πυροσβέστες, οδηγοί).

Κύριες Λειτουργίες:

- **Μέθοδος GET:**
  - Ανακτά τον τύπο του εθελοντή από τα cookies.
  - Φιλτράρει τα περιστατικά βάσει του τύπου εθελοντή (π.χ., οι οδηγοί βλέπουν περιστατικά που απαιτούν οχήματα).
  - Επιστρέφει μια JSON λίστα με τα αντίστοιχα περιστατικά.

3.2.3 Servlets του User

1. CookieServlet

Σκοπός:
Το `CookieServlet` είναι υπεύθυνο για τη διαχείριση των cookies που σχετίζονται με τον χρήστη, επιτρέποντας την αποθήκευση και ανάκτηση δεδομένων σύνδεσης.

Κύριες Λειτουργίες:

- **Μέθοδος GET:**
  - Ανακτά όλα τα cookies από το αίτημα και ελέγχει αν υπάρχει cookie με το όνομα `"userData"`.

    - Αν υπάρχει, αποκωδικοποιεί το περιεχόμενό του και το μετατρέπει σε JSON αντικείμενο.
    - Επιστρέφει μια απάντηση JSON που δηλώνει ότι ο χρήστης είναι συνδεδεμένος και περιέχει τα δεδομένα του.
    - Αν δεν βρεθεί το cookie, επιστρέφει μια απάντηση που δηλώνει ότι ο χρήστης **δεν είναι συνδεδεμένος**.
- **Μέθοδος POST:**
    - Δημιουργεί ένα κενό cookie `"userData"` και το ορίζει να λήξει άμεσα.
    - Αυτό έχει ως αποτέλεσμα τη **διαγραφή του cookie**, αποσυνδέοντας τον χρήστη.
    - Επιστρέφει μια απάντηση JSON που δηλώνει ότι η αποσύνδεση ήταν επιτυχής.

---

2. LoginServlet

Σκοπός:
Το `LoginServlet` διαχειρίζεται τη διαδικασία σύνδεσης και ενημέρωσης του προφίλ των χρηστών.

Κύριες Λειτουργίες:

- **Μέθοδος GET:**
    - Λαμβάνει το **username** και το **password** ως παραμέτρους από το αίτημα.
    - Συνδέεται στη βάση δεδομένων και ελέγχει αν υπάρχει χρήστης με αυτά τα διαπιστευτήρια.
    - Αν ο χρήστης βρεθεί, επιστρέφει ένα JSON αντικείμενο με όλα τα προσωπικά του δεδομένα.
    - Δημιουργεί επίσης ένα cookie `"userData"` που περιέχει τα στοιχεία του χρήστη, ώστε να παραμένει συνδεδεμένος.
    - Αν τα διαπιστευτήρια είναι λανθασμένα, επιστρέφει μια απάντηση που δηλώνει ότι η σύνδεση **απέτυχε**.
- **Μέθοδος POST:**
    - Λαμβάνει ένα JSON αντικείμενο με ενημερωμένα στοιχεία χρήστη.
    - Εκτελεί ένα SQL `UPDATE` στον πίνακα `users` για να ανανεώσει τις πληροφορίες του χρήστη στη βάση δεδομένων.

- Αν η ενημέρωση είναι επιτυχής, επιστρέφει μήνυμα επιτυχίας και ενημερώνει το cookie `"userData"`.
- Αν αποτύχει, επιστρέφει μήνυμα αποτυχίας.

---

3. RunningIncidentsUserServlet

Σκοπός:
Το `RunningIncidentsUserServlet` επιτρέπει στους εγγεγραμμένους χρήστες να βλέπουν τα ενεργά περιστατικά στο σύστημα.

Κύριες Λειτουργίες:

- **Μέθοδος GET:**
  - Ανακτά από τη βάση δεδομένων όλα τα περιστατικά που βρίσκονται σε κατάσταση `"running"`.
  - Τα μετατρέπει σε JSON και τα επιστρέφει ως απάντηση.
  - Αν παρουσιαστεί σφάλμα στη βάση δεδομένων, επιστρέφει μια κενή λίστα.

---

4. NewServlet

Σκοπός:
Το `NewServlet` επιτρέπει την εγγραφή νέων χρηστών στο σύστημα.

Κύριες Λειτουργίες:

- **Μέθοδος POST:**
  - Διαβάζει δεδομένα από το αίτημα JSON.
  - Χρησιμοποιεί την κλάση `EditUsersTable` για να προσθέσει έναν νέο χρήστη στη βάση δεδομένων.
  - Αν η εγγραφή είναι επιτυχής, επιστρέφει μήνυμα επιτυχίας.
  - Αν παρουσιαστεί σφάλμα, επιστρέφει μήνυμα αποτυχίας.

3.2.4 Servlets του guest

1. RunningIncidentsGuestServlet

Σκοπός:
Το `RunningIncidentsServlet` εξυπηρετεί χρήστες που δεν έχουν κάνει εγγραφή ή σύνδεση, παρέχοντας μια λίστα με όλα τα περιστατικά που βρίσκονται σε κατάσταση **"running"**. Αυτό επιτρέπει σε επισκέπτες να ενημερώνονται για έκτακτα γεγονότα, χωρίς να απαιτείται αυθεντικοποίηση.

Κύριες Λειτουργίες:

- **Μέθοδος GET:**
    - Ορίζει το `Content-Type` της απόκρισης ως `application/json`.
    - Κάνει κλήση στη βάση δεδομένων για να ανακτήσει περιστατικά που έχουν κατάσταση **"running"**.
    - Μετατρέπει τα περιστατικά σε μορφή JSON και τα επιστρέφει στην απάντηση.
- **Μέθοδος `fetchRunningIncidentsFromDatabase()`:**
    - Πραγματοποιεί αναζήτηση στη βάση δεδομένων μέσω της κλάσης `EditIncidentsTable`.
    - Επιστρέφει όλα τα ενεργά περιστατικά.

Διαφορά από άλλα servlets:

- Δεν απαιτεί cookie αυθεντικοποίησης (`volunteerData`), όπως τα servlets για εθελοντές.
- Δεν φιλτράρει περιστατικά με βάση τον τύπο χρήστη (π.χ., οδηγός, διασώστης), αλλά επιστρέφει **όλα τα ενεργά περιστατικά**.
- Επιτρέπει την **ελεύθερη πρόσβαση** σε δεδομένα, διευκολύνοντας την ενημέρωση των επισκεπτών.

3.2.5 Κοινόχρηστα Servlets

1. MessageServlet

Το `MessageServlet` χρησιμοποιείται τόσο από διαχειριστές όσο και από χρήστες, επιτρέποντας την αποστολή και λήψη μηνυμάτων που σχετίζονται με ένα περιστατικό.

Κύριες Λειτουργίες

Μέθοδος GET:

- Λαμβάνει δύο παραμέτρους:
  - `username`: το όνομα χρήστη που ζητά τα μηνύματα.
  - `incident_id`: το αναγνωριστικό του περιστατικού στο οποίο αναφέρονται τα μηνύματα.
- Αν ο χρήστης **δεν είναι admin**, ανακτά τα μηνύματα που είναι ορατά μόνο σε αυτόν (`getMessagesByUser`).
- Αν ο χρήστης **είναι admin**, επιστρέφει όλα τα μηνύματα του περιστατικού (`databaseToMessage`).
- Επιστρέφει τα δεδομένα σε μορφή JSON.
- Αν προκύψει σφάλμα, επιστρέφει μήνυμα αποτυχίας.

Μέθοδος POST:

- Διαβάζει το **JSON αίτημα** που περιέχει ένα νέο μήνυμα.
- Χρησιμοποιεί την `EditMessagesTable` για να αποθηκεύσει το μήνυμα στη βάση δεδομένων.
- Ορίζει την ημερομηνία και ώρα αποστολής του μηνύματος (`setDate_time`).
- Επιστρέφει μήνυμα επιτυχίας αν το μήνυμα καταχωρηθεί σωστά.
- Αν παρουσιαστεί σφάλμα, επιστρέφει μήνυμα αποτυχίας.

3.3 Διαχείριση αιτημάτων και απαντήσεων

Η επικοινωνία μεταξύ του frontend και του backend του συστήματος **E-199** βασίζεται σε HTTP αιτήματα και απαντήσεις που ακολουθούν το πρότυπο REST. Το backend έχει αναπτυχθεί με **Java Servlets** και **Spark Java**, τα οποία διαχειρίζονται τα αιτήματα μέσω GET, POST, PUT και DELETE μεθόδων.

3.3.1 Διαχείριση αιτημάτων

Τα αιτήματα από το frontend προς το backend αποστέλλονται μέσω AJAX κλήσεων, χρησιμοποιώντας **Fetch API** ή **XMLHttpRequest**. Κάθε αίτημα περιλαμβάνει δεδομένα σε μορφή **JSON**, ώστε να διασφαλίζεται η εύκολη μετατροπή και επεξεργασία τους από τις Java κλάσεις του backend.

Οι βασικοί τύποι αιτημάτων περιλαμβάνουν:

- **GET**: Ανάκτηση δεδομένων (π.χ., λίστα περιστατικών, στοιχεία χρήστη).
- **POST**: Δημιουργία νέων εγγραφών στη βάση (π.χ., καταχώρηση νέου περιστατικού).
- **PUT**: Ενημέρωση υπαρχόντων εγγραφών (π.χ., αλλαγή κατάστασης περιστατικού).
- **DELETE**: Διαγραφή εγγραφών από τη βάση (π.χ., αφαίρεση συμμετοχής εθελοντή).

Κάθε αίτημα περιλαμβάνει **headers** για την επικοινωνία, όπως `Content-Type: application/json` και, σε περιπτώσεις ελεγχόμενης πρόσβασης, **authentication tokens** για την επικύρωση της ταυτότητας του χρήστη.

3.3.2 Διαχείριση απαντήσεων

Το backend επιστρέφει απαντήσεις στις κλήσεις του frontend με HTTP status codes και δεδομένα σε μορφή **JSON**. Οι απαντήσεις ακολουθούν μια τυποποιημένη δομή που περιλαμβάνει:

- **status**: Τον κωδικό κατάστασης HTTP (`200 OK`, `400 Bad Request`, `401 Unauthorized`, `500 Internal Server Error`).
- **message**: Περιγραφή της κατάστασης του αιτήματος (π.χ., "Login successful", "Invalid parameters").

- **data**: Τα δεδομένα που επιστρέφονται (π.χ., λίστα περιστατικών, πληροφορίες χρήστη).

Παραδείγματα απαντήσεων:

```
{
  "status": 200,
  "message": "Incident retrieved successfully",
  "data": {
    "incident_id": 101,
    "incident_type": "Fire",
    "location": "Athens, Greece"
  }
}
```

Σε περιπτώσεις σφαλμάτων, το backend στέλνει **κατάλληλους κωδικούς HTTP** και επεξηγηματικά μηνύματα σφάλματος για διευκόλυνση του debugging. Ενδεικτικά:

- `400 Bad Request`: Αποστέλλεται όταν υπάρχουν ελλιπή ή λανθασμένα δεδομένα στο αίτημα.
- `401 Unauthorized`: Χρησιμοποιείται όταν ένας μη εξουσιοδοτημένος χρήστης προσπαθεί να αποκτήσει πρόσβαση σε προστατευμένα δεδομένα.
- `500 Internal Server Error`: Δηλώνει πρόβλημα στον server ή τη βάση δεδομένων.

3.3.3 Middleware και διαχείριση συνεδριών

Για τη διαχείριση της ταυτότητας των χρηστών και την ασφάλεια, χρησιμοποιούνται **cookies** και **session tokens**. Οι κλήσεις από το frontend περιλαμβάνουν tokens τα οποία ελέγχονται από τα servlets του backend, διασφαλίζοντας ότι μόνο εγγεγραμμένοι χρήστες μπορούν να αλληλεπιδράσουν με το σύστημα.

Το backend χρησιμοποιεί middleware για:

- **Έλεγχο εξουσιοδότησης**: Επιβεβαιώνει αν ένας χρήστης έχει τα απαραίτητα δικαιώματα για να εκτελέσει μια ενέργεια.
- **Logging**: Καταγραφή αιτημάτων και απαντήσεων για λόγους παρακολούθησης και debugging.
- **Rate limiting**: Περιορισμός συχνών αιτημάτων από τον ίδιο χρήστη για προστασία από επιθέσεις DDoS.

Με αυτή τη δομή, διασφαλίζεται η αποτελεσματική και ασφαλής επικοινωνία μεταξύ των πελατών και του διακομιστή του συστήματος **E-199**, παρέχοντας γρήγορη απόκριση και αξιόπιστη επεξεργασία δεδομένων.

4. Σχεδίαση του Front-end

4.1 Χρήση JavaScript Libraries και HTML

Το frontend του συστήματος **E-199** είναι σχεδιασμένο με HTML, CSS και JavaScript, με έμφαση στη διαδραστικότητα και την εύκολη πρόσβαση στις πληροφορίες για τους χρήστες.

Για την ανάπτυξη του UI χρησιμοποιούνται οι εξής βιβλιοθήκες και τεχνολογίες:

- **Leaflet.js**: Χρησιμοποιείται για την απεικόνιση των περιστατικών στον χάρτη μέσω OpenStreetMap.
- **AJAX (XMLHttpRequest / Fetch API)**: Χρησιμοποιείται για την ασύγχρονη επικοινωνία με το backend.
- **Bootstrap**: Βελτιώνει το σχεδιασμό των HTML σελίδων μέσω έτοιμων CSS και JS components.
- **JavaScript DOM Manipulation**: Για την ενημέρωση των στοιχείων της σελίδας χωρίς πλήρη ανανέωση.

4.2 Ρόλος των CSS αρχείων

Το σύστημα περιλαμβάνει διάφορα CSS αρχεία για τον διαχωρισμό των στυλ μεταξύ των διαφορετικών σελίδων:

- **index.css**: Βασικό στυλ της αρχικής σελίδας.
- **incidentSTYLE.css**: Περιλαμβάνει κανόνες μορφοποίησης για τη σελίδα καταγραφής περιστατικών.
- **map.css**: Προσαρμόζει την εμφάνιση του διαδραστικού χάρτη Leaflet.js.
- **login.css, loginAdmin.css, loginVolunteerPage.css**: Προσαρμόζουν το UI των σελίδων σύνδεσης.
- **volunteer_incidents.css**: Στυλ για τη διαχείριση των περιστατικών από την πλευρά των εθελοντών.

Τα CSS αρχεία οργανώνονται ώστε να επιτρέπουν ένα συνεκτικό και προσαρμόσιμο UI. Επιπλέον, χρησιμοποιούνται media queries για τη διατήρηση της responsive συμπεριφοράς της εφαρμογής, εξασφαλίζοντας ότι η εμφάνιση παραμένει λειτουργική και αισθητικά ευχάριστη σε διαφορετικές συσκευές.

Επίσης, γίνεται εκτεταμένη χρήση μεταβλητών CSS για την εύκολη προσαρμογή χρωμάτων, γραμματοσειρών και άλλων σχεδιαστικών χαρακτηριστικών, διατηρώντας έτσι μια ενιαία αισθητική σε όλο το σύστημα.

4.3 Client-side λογική με JavaScript

Η διαχείριση των χρηστών και των περιστατικών γίνεται μέσω JavaScript αρχείων που χειρίζονται τα αντίστοιχα events και AJAX αιτήματα:

- **index.js**: Χειρίζεται την αρχική φόρτωση της σελίδας, διαχειρίζεται τα βασικά στοιχεία του UI και προετοιμάζει event listeners για αλληλεπιδράσεις των χρηστών.
- **incidentAJAX.js**: Διαχειρίζεται τις AJAX κλήσεις για την καταγραφή και ενημέρωση περιστατικών, επιτρέποντας στους χρήστες να δημιουργούν και να επεξεργάζονται περιστατικά μέσω του REST API.
- **loginAJAX.js, loginAdminAJAX.js, loginVolunteerAJAX.js**: Υπεύθυνα για τις διαδικασίες σύνδεσης των χρηστών (πολίτες, διαχειριστές, εθελοντές). Περιλαμβάνουν κλήσεις προς το backend για τον έλεγχο των credentials και τη διαχείριση των cookies συνεδρίας.
- **htmlRedirections.js**: Περιέχει συναρτήσεις που διευκολύνουν τη μετάβαση μεταξύ σελίδων, όπως δυναμικά redirects ανάλογα με το ρόλο του χρήστη ή τη διατήρηση προηγούμενων states μέσω URL parameters.
- **manicipality.js**: Επεξεργάζεται δεδομένα σχετιζόμενα με δήμους και περιοχές, επιτρέποντας την προεπιλογή περιοχών κατά την καταχώρηση ενός περιστατικού.
- **volunteer_incidents.js**: Διαχειρίζεται την προβολή και συμμετοχή των εθελοντών σε περιστατικά, ανακτώντας από το backend περιστατικά που σχετίζονται με τον συγκεκριμένο εθελοντή και επιτρέποντας την αποδοχή ή απόρριψη συμμετοχής.
- **map.js**: Διαχειρίζεται την χαρτογράφηση των περιστατικών μέσω Leaflet.js, φορτώνοντας δεδομένα περιστατικών από το backend, εμφανίζοντας markers στον χάρτη και παρέχοντας λειτουργίες όπως zoom, κλικ για εμφάνιση πληροφοριών και δυναμική ενημέρωση δεδομένων.

Όλα τα παραπάνω αρχεία οργανώνονται ώστε να διαχωρίζουν τη λογική των λειτουργιών, βελτιώνοντας την επεκτασιμότητα και τη συντήρηση του κώδικα. Ειδικότερα, δίνεται έμφαση στη modular ανάπτυξη, όπου κάθε αρχείο JavaScript

αναλαμβάνει έναν συγκεκριμένο ρόλο, αποφεύγοντας την αλληλοεξάρτηση και διευκολύνοντας την τροποποίηση ή την επέκταση λειτουργιών.

Επιπλέον, υλοποιείται μηχανισμός caching για τα δεδομένα των περιστατικών, μειώνοντας τα αιτήματα προς το backend και βελτιώνοντας την ταχύτητα απόκρισης της εφαρμογής. Για παράδειγμα, τα δεδομένα αποθηκεύονται τοπικά στο session storage, ώστε να είναι διαθέσιμα ακόμα και σε περιπτώσεις προσωρινής αποσύνδεσης από το διαδίκτυο.

5. Ασύγχρονη Λειτουργία (AJAX & REST Requests)

5.1 Περιγραφή AJAX λειτουργιών

Η ασύγχρονη επικοινωνία στο σύστημα **E-199** βασίζεται στην τεχνολογία **AJAX (Asynchronous JavaScript and XML)**, επιτρέποντας την αποστολή και λήψη δεδομένων από το backend χωρίς την ανάγκη πλήρους ανανέωσης της σελίδας.

Το AJAX χρησιμοποιείται για:

- **Δυναμική φόρτωση δεδομένων**: Οι χρήστες λαμβάνουν πληροφορίες χωρίς να απαιτείται ανανέωση της σελίδας.
- **Ενημέρωση περιεχομένου σε πραγματικό χρόνο**: Τα περιστατικά ανανεώνονται αυτόματα.
- **Αποστολή δεδομένων στο backend**: Τα αιτήματα αποστέλλονται μέσω JSON φορμάτ για άμεση επεξεργασία.
- **Διαχείριση χρηστών και συνεδριών**: Οι συνδέσεις και αποσυνδέσεις πραγματοποιούνται ασύγχρονα.

Η βασική υλοποίηση AJAX στο E-199 χρησιμοποιεί **Fetch API**, αλλά υπάρχουν επίσης περιπτώσεις όπου χρησιμοποιείται το κλασικό `XMLHttpRequest (XHR)`.

Η παραπάνω κλήση λαμβάνει δεδομένα περιστατικών από το backend και τα επεξεργάζεται δυναμικά.

Παράδειγμα AJAX αίτησης με XHR

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        console.log(JSON.parse(this.responseText));
    }
};
xhttp.open("GET", "/localhost8081/incidents", true);
xhttp.send();
```

Αυτή η μέθοδος χρησιμοποιεί `XMLHttpRequest` για να ανακτήσει δεδομένα, προσφέροντας μεγαλύτερο έλεγχο στις κλήσεις AJAX.

5.2 REST API στο Backend

Το backend του **E-199** βασίζεται σε **RESTful API**, όπου τα endpoints ανταποκρίνονται σε αιτήματα HTTP. Οι βασικές λειτουργίες που υποστηρίζει περιλαμβάνουν:

- `GET /incidents` – Ανάκτηση όλων των περιστατικών.
- `POST /incidents` – Δημιουργία νέου περιστατικού.
- `PUT /incidents/{id}` – Ενημέρωση υπάρχοντος περιστατικού.
- `DELETE /incidents/{id}` – Διαγραφή περιστατικού.

Η επικοινωνία γίνεται με **JSON φορμάτ**, διευκολύνοντας τη μεταφορά και ανάλυση των δεδομένων.

5.3 Διαχείριση Απαντήσεων και Σφαλμάτων

Η διαχείριση των αποκρίσεων γίνεται με τη χρήση **HTTP Status Codes**:

- `200 OK` – Επιτυχής απόκριση.
- `400 Bad Request` – Λανθασμένα δεδομένα στο αίτημα.
- `401 Unauthorized` – Χρήστης χωρίς δικαιώματα πρόσβασης.
- `500 Internal Server Error` – Σφάλμα στον server.

Αν μια αίτηση περιέχει σφάλματα, το backend επιστρέφει μήνυμα σφάλματος:

```
{
    "status": 400,
    "message": "Invalid request parameters"
}
```

5.4 One-Page Application και API Επικοινωνία

Το σύστημα **E-199** έχει υλοποιηθεί ακολουθώντας την αρχιτεκτονική **One-Page Application (SPA)**, όπου το frontend διαχειρίζεται δυναμικά τις αλλαγές περιεχομένου χωρίς πλήρη ανανέωση της σελίδας.

- **Διαχείριση Σελίδων μέσω JavaScript**: Οι χρήστες μπορούν να μετακινούνται μεταξύ διαφορετικών τμημάτων της εφαρμογής μέσω JavaScript, χωρίς ανάγκη για νέες φόρτωση HTML σελίδων από τον server.
- **AJAX και Fetch API**: Όλα τα δεδομένα ανακτώνται και αποστέλλονται μέσω AJAX, διασφαλίζοντας ότι οι αλληλεπιδράσεις με το backend είναι άμεσες και ασύγχρονες.
- **State Management με Local Storage και Session Storage**: Αποθήκευση δεδομένων τοπικά για μείωση των αιτημάτων προς τον server.

6. Κατηγορίες και Διαχείριση Περιστατικών

Το σύστημα **E-199** υποστηρίζει την καταγραφή και διαχείριση διαφόρων τύπων περιστατικών έκτακτης ανάγκης, επιτρέποντας στις αρχές και στους εθελοντές να ανταποκρίνονται έγκαιρα. Κάθε περιστατικό περιλαμβάνει βασικά δεδομένα, όπως τύπο συμβάντος, τοποθεσία, επίπεδο επικινδυνότητας και χρονικά δεδομένα.

6.1 Κατηγορίες Περιστατικών

Τα περιστατικά που μπορούν να καταχωρηθούν στο σύστημα ανήκουν στις εξής κατηγορίες:

- **Πυρκαγιές**: Συμπεριλαμβάνουν αστικές και δασικές πυρκαγιές.
- **Τροχαία ατυχήματα**: Καταγραφές συγκρούσεων οχημάτων και ατυχημάτων στον δρόμο.
- **Πλημμύρες**: Περιπτώσεις πλημμυρικών φαινομένων που επηρεάζουν κατοικημένες περιοχές.
- **Σεισμοί**: Καταγραφή ζημιών ή κινδύνων που προκλήθηκαν από σεισμικές δονήσεις.
- **Ιατρικά περιστατικά**: Επείγοντα περιστατικά που αφορούν την υγεία πολιτών, όπως λιποθυμίες ή καρδιακά επεισόδια.
- **Εγκληματικές ενέργειες**: Συμβάντα που αφορούν εγκληματικές πράξεις, όπως διαρρήξεις ή επιθέσεις.
- **Τεχνικές βλάβες**: Διακοπές ρεύματος, ατυχήματα σε υποδομές κ.λπ.

6.2 Καταγραφή και Παρακολούθηση Περιστατικών

Η διαδικασία καταγραφής και παρακολούθησης ενός περιστατικού ακολουθεί συγκεκριμένα βήματα:

1. **Δημιουργία Περιστατικού**: Ο χρήστης ή ένας φορέας καταχωρεί το περιστατικό μέσω της πλατφόρμας, συμπληρώνοντας όλα τα απαιτούμενα πεδία.
2. **Έλεγχος και Έγκριση**: Οι διαχειριστές του συστήματος επιβεβαιώνουν την εγκυρότητα του περιστατικού.
3. **Ανάθεση σε Εθελοντές**: Αν το περιστατικό απαιτεί δράση, εθελοντές ή αρχές ενημερώνονται και καλούνται να ανταποκριθούν.

4. **Ενημερώσεις και Διαχείριση Κατάστασης**: Το περιστατικό ενημερώνεται δυναμικά με νέα δεδομένα και η κατάσταση μπορεί να αλλάζει (π.χ., από "ενεργό" σε "υπό έλεγχο" ή "ολοκληρωμένο").
5. **Αρχείο και Στατιστικά**: Μετά την ολοκλήρωση του περιστατικού, τα δεδομένα αποθηκεύονται για ανάλυση και βελτίωση της διαχείρισης κρίσεων.

7. API Χρήση & Επικοινωνία με Backend

7.1 Εσωτερικά APIs

Το σύστημα **E-199** χρησιμοποιεί ένα σύνολο εσωτερικών APIs για την επικοινωνία μεταξύ του frontend και του backend. Αυτά τα APIs είναι υπεύθυνα για τη διαχείριση περιστατικών, χρηστών, εθελοντών και στατιστικών δεδομένων.

Τα βασικά endpoints των εσωτερικών APIs περιλαμβάνουν:

- `GET /api/incidents` – Ανάκτηση όλων των περιστατικών.
- `POST /api/incidents` – Δημιουργία νέου περιστατικού.
- `PUT /api/incidents/{id}` – Ενημέρωση υπάρχοντος περιστατικού.
- `DELETE /api/incidents/{id}` – Διαγραφή περιστατικού.
- `GET /api/users/{id}` – Ανάκτηση δεδομένων συγκεκριμένου χρήστη.
- `POST /api/login` – Αυθεντικοποίηση χρήστη και δημιουργία συνεδρίας.
- `GET /api/messages/{incident_id}` – Λήψη μηνυμάτων σχετικών με ένα περιστατικό.
- `POST /api/messages` – Αποστολή νέου μηνύματος μεταξύ χρηστών ή εθελοντών.

Όλα τα αιτήματα και οι αποκρίσεις ακολουθούν το JSON format, διασφαλίζοντας την εύκολη διαχείριση και ανάλυση των δεδομένων στο frontend.

7.2 Εξωτερικά APIs

Εκτός από τα εσωτερικά APIs, το **E-199** χρησιμοποιεί και εξωτερικές υπηρεσίες API για να βελτιώσει τη λειτουργικότητα του συστήματος. Αυτά τα APIs περιλαμβάνουν:

- **Leaflet.js / OpenStreetMap API**: Χρησιμοποιείται για την απεικόνιση των περιστατικών σε διαδραστικό χάρτη.
- **Google Geocoding API**: Επιτρέπει τη μετατροπή διευθύνσεων σε γεωγραφικές συντεταγμένες.

Οι εξωτερικές αυτές υπηρεσίες ενσωματώνονται μέσω AJAX αιτημάτων και βοηθούν στην επέκταση της λειτουργικότητας του συστήματος, επιτρέποντας στους χρήστες να λαμβάνουν περισσότερες πληροφορίες σχετικά με τα περιστατικά.