**CRISIS CONTAINMENT SERVICE (CriC)**

1. **Descrierea aplicatiei**

Crisis Containment Service (CriC) reprezinta o aplicatie Web care ofera posibilitatea avertizarii in timp real a cetatenilor si mobilizarii autoritatilor in legatura cu situatiile de urgenta precum incendii, cutremure, inundatii, alunecari de teren petrecute in diferite zone. Site-ul va furniza/va permite introducerea de informatii necesare privind situatiile de criza si zonele afectate, acestea din urma fiind individualizate in functie de tipul si intensitatea dezastrului natural. Daca in urma dezastrelor se intampla ca unii cetateni sa nu mai dea de urma unor membrii ai familiei sau apropiati, acest site va permite introducerea de informatii in legatura cu persoanele disparute sau gasite.

Aplicatia CriC utilizeaza serviciile Google Crisis Map si Fusion Tables care furnizeaza in timp real informatiile necesare privind situatiile de criza si zonele afectate, acestea din urma fiind individualizate diferit in functie de tipul dezastrului natural si de intensitatea acestuia.

Aditional, in cadrul aplicatiei se va utiliza serviciul CAP (Common Alerting Protocol) pentru transmiterea de notificari (prin intermediul media, a platformelor sociale: Facebook, Instagram, etc) si alerte populatiei (afectate direct sau aflate in proximitatea zonei cu risc).

1. **Structura aplicatiei**

Aplicatia CriC este structurata astfel:

1. **ACASA** – reprezinta pagina principala a aplicatiei care contine un formular ce permite utilizatorului sa se logheze. In ipoteza logarii cu succes, utilizatorul are dreptul de a adauga pe harta zonele afectate de evenimentele naturale. In caz de esec, acesta are la dispozitie mai multe incercari pentru a se loga.
2. **EVENIMENTE** – pagina care afiseaza o harta completata utilizand serviciul Google Fusion Table si in care se pot adauga evenimente. Aceasta este conectata cu tabelele Fusion Table direct din contul Google.
3. **LOCALIZARE EVENIMENTE** – pagina este vizibila doar pentru userii logati si permite adaugarea evenimentelor pe harta.

Pagina contine o serie de inputuri reprezentand zona afectata, tipul de eveniment, aria afectata etc. Inputurile sunt trimise catre serverul Google. Inainte de a insera in tabelul Fusion Table, Google verifica daca serverul nostru are dreptul sa insereze in tabelul in care sunt stocate informatiile.

Tabelul (asemanator unui tabel din baza de date) in care sunt stocate informatiile este salvat in Google Drive. Din acesta isi extrage informatiile Crisis Map.

1. **PERSOANE DISPARUTE** - pagina vizibila pentru vizitatori. Aici exista doua posibilitati:

a) ***Cauta o persoana*** – se introduce numele si prenumele acesteia. Datele sunt cautate intr-un repository Google Person Finder. Accesarea repositori-ului se face prin intermediul metodei Get, care solicita serverului Google sa caute persoana respectiva. Serverul returneaza un XML cu informatii despre acea persoana, pe care il parsam.

b) ***Detin informatii despre o persoana***. Se genereaza un document XML dupa un format care se gaseste in documentatia de la Person Finder. Documentul contine informatii luate din inputul trimis de un utilizator. Se returneaza tot un XML care confirma ca datele au fost inserate sau nu. Pentru trimiterea XML-ului se foloseste metoda Post.

1. **SETARI** – pagina permite utilizatorului sa isi reseteze parola si informatiile personale. De asemenea, aceasta permite administratorului sa stearga utilizatorii.
2. **INREGISTRARE** – pagina adauga un utilizator in baza de date. Permite si logarea (exact ca in pagina Acasa).
3. **TRIMITE ALERTA** – pagina disponibila doar pentru administrator. Prin intermediul acesteia, se pot trimite alerte atat pe email cat si pe mobil (este necesara instalarea aplicatiei mobile pentru a putea trimite astfel de alerte).

Alertele sunt trimise catre un numar de persoane care exista deja intr-un tabel.

1. **CONTACT** – pagina ce afiseaza datele de contact (informatii generale)

1. **Utilizatorii aplicatiei**
2. **Administratorul**

* Logare
* Gestionare eveniment (vizualizare/ introducere/ stergere)
* Gestionare baza de date (vizualizare/ stergere/ modificare rang utilizatori)
* Cautare/ Introducere persoana disparuta
* Alertare populatie
* Delogare

1. **Utilizatorul**

* Inregistrare
* Logare
* Gestionare eveniment (vizualizare/ introducere)
* Cautare/ Introducere persoana disparuta
* Editare pagina Setari (Profil)
* Delogare

1. **Vizitatorul**

* Cautare/ Introducere persoana disparuta
* Vizualizare eveniment

1. **Fusion Table API**

* Inserare intr-un tabel date despre un eveniment
* Stergere evenimente

1. **Crisis Map**

* Afisare eveniment

1. **Person Finder API**

* Furnizare informatii despre persoana disparuta
* Permite introducerea unor date

1. **SMS Gateway API**

* Trimitere mesaje de alertare

1. **Baza de date**

* Retine informatii despre utilizatori

1. **Tehnologiile utilizate**

***a)* *Client:***

* **HTML –** pentru a reda continutul paginii
* **CSS –** pentru a oferi stilizare continutului

1. ***Server:***

* **PHP –** utilizat pentru partea de backend deoarece este un limbaj usor de folosit, gandit pentru WEB si care permite conectarea usoara la baza de date
* **Baze de date: MySQL** usor de utilizat cu PHP. Este o baza de date relationala usor de administrat. Nu folosim baza de date NoSQL deoarece nu este potrivita pentru aplicatia creata. NoSQL este utila atunci cand avem nevoie de flexibilitate. In cazul de fata nu e nevoie de flexibiltate ci este nevoie de doua tabele care sa retina utilizatorii si persoanele pentru alertare.
* **Google Fusion Table -** pentru folosirea acestuia a fost nevoie si de OAuth 2.0 care permite comunicarea dintre serverul nostru si serverul Google pe baza unui fisier JSON generat de Google, care a fost introdus in aplicatia noastra si care informeaza serverul Google ca aplicatia poate sa foloseaca serviciul. A fost nevoie si de Google PHP client library pentru a conecta cele doua servere.
* **Google Person Finder -** cautarea de informatii se face prin metoda GET. URL-ul primeste ca parametri o interogare, iar serverul Google returneaza un XML
* adaugarea informatiilor – format XML, completat cu informatii despre persoana disparuta si trimis catre server prin metoda POST. Serverul returneaza un raspuns sub forma de XML.
* **Common Alerting Protocol (CAP)** constituie un protocol comun de trimitere a alertelor (CAP), reprezentat de date în format XML, utilizat pentru schimbul de avertismente și urgențe între tehnologiile de alertare. CAP permite ca un mesaj de avertizare să fie difuzat în mod constant în mai multe aplicații, simultan cu multe sisteme de avertizare, cum ar fi Google Public Alerts. PAC sporește eficacitatea avertizării și simplifică modalitatea activării unui avertisment pentru cei responsabili.
* **PHPMailer** – reprezinta cel mai popular cod pentru trimiterea de emailuri din PHP. Beneficiaza de suport integrat SMTP, oferind posibilitatea de a trimite emailuri pe platforme Windows fara un server de mail local. PHPMailer poate trimite emailuri cu multiple adrese TO, CC, BCC and Reply-to. Codul valideaza in mod automat adresele de email si le protejeaza impotriva atacurilor.
* **SMS Gateway** –Acest serviciu se poate utiliza pentru a trimite in mod planificat un mesaj prin intermediul telefoanelor ce utilizeaza SO Android. Se creeaza un JSON care contine informatiile din mesajele SMS si pe care API trebuie sa le trimita mai departe.

1. **Arhitectura**

**Model-View-Controller** este utilizat în mod obișnuit pentru dezvoltarea de software care împarte o aplicație în trei părți interconectate. Aceasta se face pentru a separa reprezentările interne ale informațiilor de modalitățile în care informațiile sunt prezentate și acceptate de la utilizator.

Modelul de proiectare MVC decuplează aceste componente majore, permițând reutilizarea eficientă a codului și dezvoltarea paralelă.

**Model** - Componenta Model este conforma cu logica legata de date cu care lucrează utilizatorul. Aceasta poate reprezenta fie datele transferate între componentele View și Controller, fie orice alte date legate de logica de afaceri.

**View -** Componenta View este utilizată pentru întreaga logică UI a aplicației.

**Controller -** Controlerele acționează ca o interfață între componentele Model și View pentru a procesa toate logica de afaceri și cererile primite, pentru a manipula datele utilizând componenta Model și a interacționa cu componenta View pentru a da rezultatul final.

In cazul aplicatiei noastre cele trei componente sunt reprezentate astfel:

**Controllerul** este un intermediar intre View si Model. Controlerul este compus din mai multe clase care extind o clasa de baza numita Controller, Aceste clase contin metode pentru a reda view-uri sau pentru a trimite date catre model, unde vor fi procesate.

**Modelul** este compus din mai multe clase care extind clasa de baza Model. Fiecare clasa extinsa are un corespondent in Controller, cu care comunica. Modelul este responsabil de logica aplicatiei, se conecteaza la baza de date si prelucreaza informatiile primite de la controler.

**View-ul** este format dintr-o singura clasa care este instantiata din Controller. Clasa contine o metoda care include si reda (afiseaza) o anumita pagina.

1. **Stabilirea taskurilor de backend pentru fiecare membru al echipei**
2. Luca Iustin - Fusion table
3. Savastre Andreea Aurora – Person Finder
4. Ghimp Sergiu – Logare, Baza de date
5. Schipor Mihaela – Alertarea.