UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL DE CALCULATOARE

PROIECT SASPS

Studiu asupra folosirii unei arhitecturi serverless în cadrul unei platforme de comerț electronic

Năstase Ștefan-Iulian

BUCUREȘTI
2023

CUPRINS

1 INTRODUCERE

1.1 Context

Comerțul electronic a devenit un element central al economiei moderne, iar evoluția rapidă a tehnologiei a adus la dezvoltarea unor abordări noi în gestionarea infrastructurii și resurselor asociate. În acest context, adoptarea unei arhitecturi serverless în cadrul platformelor de comerț electronic devine o temă de cercetare semnificativă. Arhitectura serverless oferă posibilitatea de a gestiona eficient resursele, de a asigura scalabilitatea și de a reduce costurile operaționale.

Platformele de comerț electronic se confruntă cu provocări precum gestionarea fluctuațiilor bruște ale traficului și asigurarea performanței constante, în special în cadrul evenimentelor precum Black Friday. În acest studiu, ne propunem să explorăm modul în care arhitectura serverless poate contribui la soluționarea acestor provocări, oferind flexibilitate, scalabilitate și eficiență operațională.

Odată cu trecerea la o arhitectură serverless, se deschid o serie de întrebări și aspecte practice legate de implementare, performanță și securitate. Analiza acestor aspecte va oferi o înțelegere mai profundă a impactului acestei arhitecturi asupra mediului de comerț electronic și va oferi baza pentru luarea deciziilor informate în ceea ce privește adoptarea sau ajustarea acestei tehnologii în contextul specific al fiecărei platforme.

Prin intermediul acestui studiu, ne propunem să aducem lumină asupra potențialului arhitecturii serverless în îmbunătățirea eficienței și performanței platformelor de comerț electronic, contribuind astfel la dezvoltarea și optimizarea continuă a acestui sector într-o eră digitală în continuă schimbare.

1.2 Scopul studiului

Studiul propus are drept obiectiv principal analiza și compararea performanței arhitecturii serverless cu o arhitectură tradițională bazată pe server în cadrul unei platforme de comerț electronic. Într-un mediu de comerț electronic dinamic și în continuă schimbare, performanța reprezintă un aspect esențial, influențând direct experiența utilizatorilor.

În mod specific, obiectivele studiului sunt:

- Evaluarea eficienței operaționale: Comparația costurilor operaționale asociate cu implementarea și întreținerea arhitecturii serverless versus arhitectura server-based.

- Analiza scalabilității: Evaluarea modului în care fiecare arhitectură răspunde la cerințele de scalabilitate, în special în situații de vârf ale traficului.
- Măsurarea timpului de răspuns: Compararea timpului de răspuns al platformei în condiții variate de utilizare, inclusiv în încărcături crescute.
- Testarea performanței sub sarcină: Evaluarea rezilienței și performanței sistemului în situații de trafic intens și cerințe ridicate.

Prin concentrarea pe aceste obiective, studiul va oferi o viziune detaliată a modului în care arhitectura serverless poate aduce beneficii semnificative sau, eventual, poate întâmpina provocări în comparație cu o abordare tradițională bazată pe servere în cadrul unei platforme de comerț electronic.

2 ARHITECTURA SERVERLESS: CONCEPTE DE BAZĂ

2.1 Ce este arhitectura serverless?

Arhitectura serverless reprezintă o paradigmă de dezvoltare software în care gestionarea infrastructurii și a resurselor este externalizată către furnizorii de servicii cloud, permitând dezvoltatorilor să se concentreze exclusiv pe codul și funcționalitățile aplicației, fără a fi nevoie să gestioneze direct serverele ce rulează aplicația.

Principalele caracteristici ale arhitecturii serverless includ:

1. **Lipsa serverelor:**
   Eliminarea necesității de a gestiona, scala și întreține infrastructura serverelor fizice sau virtuale.
2. **Facturare bazată pe consum:**
   Costurile sunt asociate direct cu resursele utilizate și timpul de execuție al funcțiilor, oferind un model de facturare mai granular.
3. **Scalabilitate automată:** Capacitatea de a se scala automat în funcție de volumul de cereri, asigurând performanța optimă în orice condiții de trafic.
4. **Model de execuție bazat pe evenimente:** Funcțiile serverless sunt declanșate de evenimente, cum ar fi cereri HTTP, modificări în bazele de date sau încărcări de fișiere, facilitând o arhitectură reactivă.
5. **Deschis pentru diverse limbaje de programare:** Suport pentru multiple limbaje de programare, permițând dezvoltatorilor să aleagă tehnologiile potrivite pentru proiectele lor.

În contextul unui mediu de comerț electronic, arhitectura serverless poate aduce beneficii semnificative prin adaptabilitatea sa la variațiile de trafic, eficiența în utilizarea resurselor și

eliminarea nevoii de gestionare a infrastructurii, permițând astfel concentrarea resurselor pe îmbunătățirea funcționalităților și experienței utilizatorilor.

2.2 Avantaje și dezavantaje ale arhitecturii serverless

Implementarea unei arhitecturi serverless în cadrul unei platforme de comerț electronic aduce cu sine o serie de avantaje și dezavantaje:

Avantaje

- Scalabilitate dinamică: Arhitectura serverless permite o scalabilitate automată și dinamică în funcție de cerințele de trafic, asigurând performanța optimă în orice condiții.
- Reducerea costurilor: Modelul de facturare bazat pe consum și absența necesității de întreținere a infrastructurii pot duce la costuri operaționale mai reduse.
- Dezvoltare rapidă: Concentrarea dezvoltatorilor asupra logicii aplicației și eliminarea gestionării serverelor poate accelera ciclul de dezvoltare.
- Reactivitate la evenimente: Arhitectura serverless funcționează pe baza unui model de execuție bazat pe evenimente, facilitând reacția rapidă la schimbări în sistem, cum ar fi actualizări ale bazei de date sau acțiuni ale utilizatorilor.

Dezavantaje

- Latență inițială: În unele cazuri, există o latență inițială în activarea funcțiilor serverless, care poate afecta timpul de răspuns perceput al aplicației.
- Limitări ale duratei de execuție: Funcțiile serverless pot avea restricții asupra timpului de execuție, ceea ce poate reprezenta o provocare pentru operațiunile care necesită procesare extinsă.
- Gestionarea stării: Abordarea serverless, care favorizează funcții independente, poate complica gestionarea stării persistente între diferite componente ale aplicației.
- Dependență de furnizorii de cloud: Utilizarea arhitecturii serverless implică o strânsă legătură cu furnizorii de servicii cloud, ceea ce poate genera dependențe și restricții în ceea ce privește portabilitatea.

3 SERVERLESS ECOMMERCE PLATFORM

Pentru realizarea acestui studiu, voi implementa o platformă online de vânzări, similară cu Amazon. Platforma va permite utilizatorilor să plaseze comenzi pentru produsele disponibile

pe stoc, fara a fi nevoie ca aceștia să fie autentificați la plasarea comenzilor.

3.1 Frontend

Componenta de frontend a aplicației apare sub forma unui Single Page Application și este scrisă folosind **React.js**.

Un Single Page Application (SPA) este o aplicație web care funcționează într-o singură pagină web și încarcă dinamic conținutul pe măsură ce utilizatorul interacționează cu aplicația, fără a necesita încărcarea întregii pagini de fiecare dată când se accesează o nouă secțiune a aplicației.

Pentru a reda pagina web, codul aplicației va fi obținut de către utilizatori folosind un bucket S3 și va fi rulat de către browser-ul clientului.

Interfață Utilizator (UI)

Pentru a oferi utilizatorilor o interfață familiară, am folosit biblioteca Material UI pentru React. Pentru a naviga pe platformă, utilizatorii au la dispoziție o bara de navigare prin care pot extinde meniul de navigare, sau pot accesa coșul de cumpărături. În figura de mai jos se poate vedea una dintre paginile aplicației.

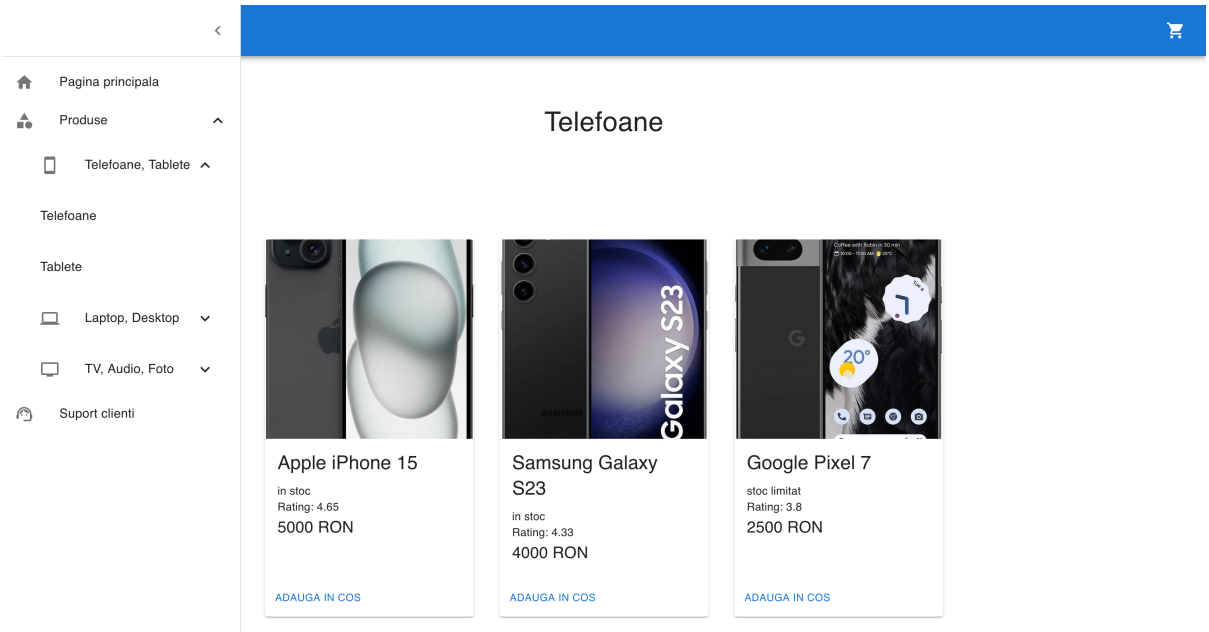

Figura 1: Pagina de produse a aplicației

Aplicația este alcătuită din 4 pagini:

- Pagina de start - include un sumar al acestui proiect;

- Pagina de produse - include o listă de produse dintr-o anumită categorie; utilizatorii pot adaugă produse în coșul de cumpărături folosind această pagină;
- Pagina de suport tehnic - include un formular prin care utilizatorii pot solicita suport tehnic;
- Pagina de cumpărături - include conținutul coșului de cumpărături, elemente de control pentru a edita conținutul coșului și un buton prin care se poate plasa o comandă; la plasarea comenzii, un formular prin care clientul își poate introduce datele va apărea.

Integrare cu componenta Backend

Pentru a interacționa cu componenta de backend a platformei, componenta de frontend va folosi **Fetch API** pentru a face apeluri la endpoint-urile expuse de către backend. La primirea răspunsului de la backend, comportamentul frontend-ului va fi influențat de status-ul răspunsului (de exemplu, daca răspunsul este 404 se va afișa un mesaj ce indică resursa lipsă).

Deoarece stocarea stărilor persistente pe backend nu este posibilă în cazul serverless, vom folosi proprietatea *localStorage* a browser-ului pentru a stoca informația necesară în cadrul aceleiași sesiuni (de exemplu, starea coșului de cumpărături).

3.2 Backend

Pentru a implementa partea de backend a aplicației serverless se vor folosi servicii oferite de către Amazon Web Services. Logica aplicatiei (business logic) va fi implementata folosind AWS Lambda si limbajul de programare Python. Ca si sursa de date pentru functille Lambda, vom folosi tabele DyanmoDB.

Business logic

Apelurile de la componenta frontend vor invoca cate o functie Lambda ce va executa functionalitatea dorita. Pentru a invoca functiile Lambda intr-un mod sincron, vom folosi functia **Lambda URLs** ce permite invocarea functiilor prin apeluri HTTP.

Surse de date

Infrastructura

4 PERFORMANȚĂ ȘI SCALABILITATE

5 CONCLUZII