SISTEME CONCURENTE ȘI DISTRIBUITE
Proiect

Echipa: Alexandra Ion, Silvia Nistor, Corina Ulariu, Dorinel Filip, George Grosu, Bogdan Mocanu

2025-2026

Cuprins

1 Cerință

În cadrul materiei Sisteme Concurente și Distribuite, fiecare student va dezvolta, sub forma unui **proiect individual**, o platformă web folosind tehnologii potrivite pentru un sistem concurent/distribuit cu performanță mare.

2 Context și motivație

Pe măsură ce aplicațiile web devin tot mai complexe și utilizatorii cer performanță ridicată, sistemele distribuite și concurente devin esențiale. Acestea permit procesarea simultană a unui număr mare de cereri, **asigură scalabilitate și continuitate a serviciilor**, precum și o gestionare eficientă a resurselor.

Acest proiect are ca scop ***familiarizarea studentului cu tehnologiile specifice sistemelor distribuite*: microservicii, containere, autentificare centralizată și rețele virtuale între servicii**. Dezvoltarea unei platforme web performante va permite aplicarea practică a conceptelor prezentate în cadrul cursurilor și a laboratorarelor, oferind experiență în proiectarea, integrarea și testarea unor soluții robuste și scalabile.

3 Detalii tehnice

3.1 Reguli generale obligatorii

Toate proiectele trebuie să respecte următoarele reguli generale:

1. **Autentificare și autorizare:** Fiecare proiect va include o funcție de autorizare utilizând o tehnologie de *Single Sign-On* (*SSO*) precum **OAuth** sau **SAML**. Implementarea se va realiza prin intermediul unui microserviciu de autentificare (ex: **Keycloak**).
2. **Managementul rolurilor:** Se va implementa o funcționalitate de gestionare a rolurilor (ex: *administrator*, *editor*, *vizitator*), care va crea automat profilurile utilizatorilor în baza meta-datelor primite de la serviciul de autentificare.
3. **Baza de date:** Fiecare proiect va conține o bază de date destinată managementului și persistenței entităților și setărilor aplicației. Integrarea bazei de date se va face prin intermediul unui **ORM** (ex: *SQLAlchemy* cu *Flask* sau *Django*).
4. **Livrarea proiectului:** Proiectul va fi livrat sub forma unui set de **servicii Docker** care vor include toate componentele necesare (inclusiv date demonstrative, dacă este cazul), astfel încât acesta să funcționeze *out-of-the-box*.
5. **Microservicii și containere:** Pentru fiecare componentă dezvoltată de către student se va include codul sursă și fișierul `Dockerfile` necesar pentru compilarea imaginii microserviciului respectiv.
6. **Structura proiectului:** Fiecare proiect va cuprinde cel puțin **5 componente**, menținând un echilibru între soluțiile *open-source* și cele dezvoltate de student (minim 2 componente proprii).
7. **Stack Docker Swarm:** Întreaga soluție va fi livrată sub forma unui **stack Docker Swarm**, în care fiecare componentă este definită ca un serviciu separat, configurat și descris în fișierul `.yml`.
8. **Interconectarea componentelor:** Comunicarea dintre servicii se va realiza prin intermediul numelor de DNS generate de Docker. Aceste valori vor fi preluate prin variabile de mediu — **nu se vor hardcoda** în cod.
9. **Rețele și securitate:** Rețelele definite vor fi configurate astfel încât fiecare serviciu să poată comunica doar cu serviciile cu care este interconectat.

10. **Replicare și testare:** Cel puțin una dintre componentele dezvoltate de student trebuie să suporte **replicare**, având o interacțiune cu o funcție avansată. Funcționarea corectă va fi demonstrată prin **unit test-uri** relevante.

Atenție! Lipsa implementării chiar și a unei singure reguli obligatorii va duce la acordarea **punctajului 0** pentru proiect.

3.2 Module și funcționalități

Fiecare proiect trebuie să expună cel puțin **5 module**, dintre care:

- 3 module sunt **comune tuturor proiectelor**:
  - Modul de autentificare;
  - Modul de profil utilizator (management-ul rolurilor);
  - Baza de date.
- 2 module se vor stabili de comun acord cu **coordonatorul proiectului**, având în vedere că acestea trebuie să fie:
  - Funcționalități **avansate**, cu implementări relevante pentru sisteme concurente sau distribuite;
  - Implementate conform bunelor practici din domeniul sistemelor concurente sau distribuite, inclusiv utilizând tehnologii și protocoale de comunicație adecvate.

4 Exemple de subiecte

Următoarele exemple respectă cerințele de arhitectură distribuită și pot fi folosite ca puncte de pornire pentru dezvoltarea proiectelor:

- **Platformă distribuită de IoT:** Expune servicii de ingestie, stocare, analiză și vizualizare a datelor folosind un **REST API**. Sistemul gestionează fluxuri mari de date provenite de la dispozitive IoT și permite monitorizare în timp real.
- **Sistem distribuit de management al comenzilor și facturilor:** Soluție pentru gestionarea comenzilor, facturilor și fluxurilor de aprobare într-o companie. Include servicii pentru managementul produselor, procesarea comenzilor, integrarea cu sisteme de plată și generarea de rapoarte de performanță.
- **Sistem de supraveghere video inteligent:** Utilizează multiple surse video și procesează imagini în timp real, incluzând detecție de obiecte, analiză de comportament și alertare automată.
- **Sistem de management al unui parc de distracții:** Gestionează clienții, rezervările și planificarea/cozile la atracții, cu posibilități de monitorizare a fluxurilor de vizitatori și gestionarea bugetelor și constrângerilor financiare.
- **Platformă distribuită de Food Delivery:** Permite utilizatorilor să vizualizeze restaurantele disponibile în zona lor geografică și să plaseze comenzi online. Include servicii de geolocalizare (**ex: Google Maps API**), managementul comenzilor, catalogul produselor per restaurant și integrarea unui sistem de plăți securizat.
- **Platformă de organizare a evenimentelor:** Permite organizatorilor să creeze și să gestioneze evenimente (concerte, piese de teatru, conferințe), iar participanților să se înscrie, să primească notificări automate și să descarce bilete PDF cu cod QR unic pentru validarea accesului.
- **Sistem de management al unei biblioteci:** Permite rezervarea și urmărirea împrumuturilor de cărți, notificarea automată la expirarea termenelor și integrarea unui mecanism de căutare distribuită (**ElasticSearch**, **OpenSearch**, **MeiliSearch**) pentru acces rapid la resurse.

5 Exemple de funcționaliți avansate

Următoarele exemple de funcționalități pot fi integrate în proiecte pentru a demonstra concepte de arhitecturi distribuite, scalabilitate și reziliență a sistemelor:

- **Sistem de rate-limiting distribuit:** Un vizitator va putea efectua cel mult *X request-uri* de un anumit tip într-un interval de timp definit. Filtrarea se va realiza într-o manieră distribuită pentru a proteja sistemul de atacuri de tip **DoS**, inclusiv atunci când cererile ajung pe noduri diferite din infrastructură.
- **Funcționalitate de caching:** Se va implementa un mecanism de *caching* pentru a evita regenerarea completă a răspunsurilor recente, atâta timp cât acestea sunt încă valide. Această abordare reduce latența și îmbunătățește performanța sistemului în condiții de trafic ridicat.
- **Sistem de monitorizare:** Va permite unui *sysop* să urmărească în timp real:
  - tipurile de request-uri procesate de sistem;
  - răspunsurile generate (de succes sau de eroare);
  - starea generală a microserviciilor.

  Se recomandă integrarea cu soluții de observabilitate precum **Prometheus**, **Grafana** sau **ELK Stack**.
- **Generarea eficientă a rapoartelor folosind replicarea bazelor de date:** Pentru a evita încărcarea bazei de date principale, se vor crea *read replicas* peste care vor fi generate rapoarte utile. Acestea pot fi expuse printr-un modul dedicat de raportare sau API REST, asigurând performanță și consistență.
- **Sistem asincron de management al plăților:** Se va implementa un microserviciu robust care gestionează plățile într-o manieră asincronă, integrând soluții externe precum **Stripe** sau **PayPal**. Comunicarea între servicii se poate realiza printr-un broker de mesaje (ex: **RabbitMQ**, **Kafka**).
- **Sistem de management al facturilor cu workeri replicați:** Funcționalitatea va permite existența mai multor *web workers* care generează facturi în paralel. Fiecare factură va avea o serie și un număr unic, garantând lipsa duplicatelor sau a numerelor lipsă în cadrul unei serii.
- **Sistem de notificare:** Utilizatorii vor primi automat email-uri sau *push notifications* bazate pe evenimente și condiții predefinite din sistem. Exemple:
  - pentru sisteme IoT – trimiterea de alerte când senzorii depășesc praguri normale;
  - notificări când un dispozitiv nu mai răspunde;
  - alerte la detectarea de anomalii în rețea.
- **Sistem de căutare rapidă:** Modulul va permite utilizatorilor să efectueze căutări eficiente în cadrul platformei, gestionând volume mari de date. Rezultatele vor fi oferite rapid prin mecanisme de indexare optimizată, folosind soluții open-source precum **ElasticSearch** sau **OpenSearch**.

6 Etape și notare

6.1 Organizare generală

- Subiectul și funcționalitățile proiectului se vor alege de comun acord cu asistentul de la laborator în **săptămâna 8**.
- Specificațiile finale ale proiectului vor fi redactate de către student și încărcate pe **Moodle** până cel târziu în **săptămâna 9** (*Milestone 1: 28 Noiembrie*). Acestea vor fi aprobate sau pot fi revizuite de către asistent până la următorul checkpoint.
- Pentru *Milestone-ul 2*, studentul trebuie să participe la laborator cel puțin o dată în intervalul **28 Noiembrie – 12 Decembrie**, iar *Milestone-ul 3* se va prezenta **înainte de 16 Ianuarie**.

- Pentru fiecare milestone, asistentul va oferi un calificativ binar (îndeplinit/neîndeplinit) ținând cont de documentația și codul sursă încărcate și prezentate de către student **până la data limită a milestone-ului**.
- Proiectul valorează 3,5 puncte din nota finală a materiei, dintre care 3 puncte se vor aloca pentru aplicația dezvoltată, iar maxim 0.5 puncte (proporțional cu punctajul obținut pentru aplicație) pentru respectarea tuturor milestone-urilor.
- Obținerea oricărei fracții din punctajul pentru respectarea tuturor milestone-urilor este condiționată de obținerea calificativului **îndeplinit** la toate cele 3 milestone-uri.
- Punctajul obținut pentru aplicație va fi stabilit în funcție de livrabilele prezentate și încărcate pentru Milestone 3.

Atenție! Orice modificare a temei sau a componenței proiectului se poate face **doar cu acordul asistentului responsabil** și va fi notată în scris de către acesta.

6.2 Structura proiectului

Proiectul se va desfășura în trei **milestone-uri**. Un milestone este considerat îndeplinit dacă rezultatele aferente sunt:

- **Încărcate** pe platforma Moodle;
- **Prezentate asistentului** de laborator până la deadline-ul asociat.

Atenție! Fiecare milestone este condiționat de îndeplinirea milestone-urilor precedente.

6.3 Milestone 1 – Alegerea proiectului (Deadline: 28 Noiembrie)

Pentru acest milestone, studentul va:

- Defini arhitectura soluției (containerele ce vor fi folosite);
- Descrie funcționalitățile platformei într-un document PDF, conform discuțiilor din prima săptămână de proiect.

6.4 Milestone 2 – Prezentare intermediară (Deadline: 12 Decembrie)

Această etapă presupune:

- Implementarea a cel puțin 40% din aplicație (funcționalitățile de bază – cele 3 obligatorii conform enunțului);
- Integrarea componentelor folosind **Docker Swarm**;
- Prezentarea stadiului proiectului pentru feedback intermediar.

6.5 Milestone 3 – Aplicația finală - 3 puncte (Deadline: 16 Ianuarie)

- Implementarea modulelor de bază (serviciul web cu funcțiile descrise în etapa 1) și prezentarea lor ca un **Stack de servicii Docker Swarm** care va include cel puțin un serviciu replicat și o bază de date – **1.2 puncte**:
  - Implementarea interfeței respectând principiile **REST API** – **0.6 puncte**;
  - Verificarea condițiilor de consistență a datelor – **0.35 puncte**;
  - Implementarea și justificarea mijloacelor de control al accesului și de securitate – **0.25 puncte**.
- Implementarea primului modul avansat – **0.9 puncte**;

- Implementarea celui de-al doilea modul avansat – **0.9 puncte**.

Respectarea tuturor milestone-urilor: 0.5 puncte, proporțional cu punctajul obținut pentru aplicație (Dacă orice termen din cele 3 nu este respectat, aceste puncte nu vor fi acordate).

Exemple:

- Punctajul final pentru un proiect care obține 2.8 puncte (din maxim 3) pentru aplicație va fi: $2.8 + 0.5 * 2.8/3.0 = 3.26$.
- Un alt proiect, cu o aplicație evaluată tot cu 2.8 puncte, dar în dezvoltarea căruia nu s-a respectat unul dintre milestone-uri, va fi notat cu 2.8 puncte, neacordându-se nicio fracție din cele 0.5 pentru respectarea milestone-urilor.

7 Recomandări de implementare și testare

Găsiți aici o serie de recomandări legate de implementarea și testarea proiectului:

1. Folosiți un repository **privat** de Git și dați commit-uri frecvent, pentru a vedea clar diferențele dintre iterațiile de cod și pentru a avea o copie sigură pe care o puteți recupera dacă ați șters ceva din greșeală sau dacă mașina pe care lucrați are probleme;
2. Dacă folosiți Python și Flask, interacțiunea cu baza de date se poate face cu **SQLAlchemy**. Alternativ, puteți folosi **Django** sau orice altă tehnologie care mapează entități din baza de date în structuri de cod.
3. **Pentru testarea API-ului și a funcționalităților implementate**, se recomandă utilizarea **Postman** și a scripturilor automate de testare.

8 Reguli privind originalitatea și proveniența rezolvărilor

Proiectul trebuie să reprezinte exclusiv munca studentului! Echipa va aplica regulamentele în vigoare pentru situațiile de fraudă / plagiat, pentru orice nerespectare a acestei reguli, incluzând, dar nelimitându-se la: copierea de la colegi (se aplică pentru toate persoanele implicate) sau din alte surse a unor părți din rezolvare, prezentarea ca rezultat personal a oricărei bucăți de cod provenită din alte surse necitate sau neaprobate în prealabil (site-uri web, alte persoane, cod generat folosit AI / LLM-uri etc.).