1. Pornind de la exemplul de implementare a agendei cu sevicii REST din laborator să se realizeze urmätoarele:
   - să se deseneze diagrama de clase;
   - să se implementeze un servlciu (utilizänd modelul portii API sau nu) care asigurä o metodä de cache ce räspunde dacä noua cerere apare in primale 30 de minute;
   Dacä se considerä necesar se mal pot crea oricäte servcii se dorese conform gändirii studentului_ Se va utiliza orchestrarea si respectarea principiilor SOLID pantru microservicii. La prazantare se va aräta diagrama de clase 9i se va explica aceasta impreunä cu gradul de respectare a principlilor_ Apol va urma discu!ia pe cod.
2. Pornind de la aplicatia exemplu de chat din laboratorul 8, să se modifice aceasta prin introducerea unui procesor de flux care implementează un mecanism de "bätaie de inimä" ce trimite mesaje false; in scopul verificärii functionärii corecte ale celorlalte microservicii. Dacä un serviciu nu functioneazä, se va trimite o cerere de replicare a acestuia unui procesor de flux dedicat Dacä se considerä necesar se mai pot crea oricäte servicii se dore$e conform gändirii studentului_ Se vor respecta principiile SOLID pentru microservicii. La prezentare se va aräta diagrama de clase se va explica aceasta impreunä cu gradul de respectare a principiilor_ Apoi va urma discutia pe cod.
3. Pornind de la aplicatia exemplu pentru ciurul lui Eratostene din laboratorul 11, să se modifice aceasta astfel incåt să se calculeze bn=suma(ai*ai), i=1 ..n. Apoi utilizånd aceastä functie in programul principal sé se calculeze pornind de Ia 100 de valori initjaljzate aleator intr-un ADT A si apoi så se depunå intr-un ADT B rezultatele pentru n=1.. 100.
4. Pornind de la aplicatia exemplu pentru ciurul lui Eratostene din laboratorul 11, să se modifice aceasta astfel fncät să se calculeze reuniunea intre A si B, unde A si B sunt douä ADT-uri ce contin fiecare cäte 100 de valori initializate aleator, apoi să se depunä intr-un ADT C rezultatele
5. Pornind de la exemplul de proiectare implementare a unei aplicatii care prime* informatii meteo (adicä utilizänd si serviciile de acolo eventual cu modificäri) să se realizeze urmätoarele:
   - un serviciu care citese dintr-un fisier (in ce format dorim) o listä de orase,
   - să se aplice modelul de proiectare de tip filtru in proiectarea implementarea unui serviciu care extrage din fluxul de date informatii pe baza numelui orasului (utilizänd serviciul anterior)
6. Pornind de Ia aplicatia exemplu cu Okazii (simulat evident) din laboratorul 7. Să se modifice aceasta astfel incåt să se introducä un procesor de flux care så realizeze statistici cu privire la erorile tipul acestora care au apärut (dacä au apärut) pentru o licitalie pänä Ia adjudecare apoi så scrie aceste informatii intr-un fl*ier local. Dacä se considerä necesar se mai pot crea oricäte servicii se doreite conform gåndirii studentului. Se vor respecta principiile SOLID pentru microserwcii, La prezentare se va aräta diagrama de clase ii se va explica aceasta impreunå cu gradul de respectare a principiilor_ Apoi va urma disculia pe cod.
7. Să se creeze in Kotlin un server TCP care să facă o cerere (khttp) pentru simbolurile compamilor cu actiuni (vezi documentatia_https://finnhub.io/docs/api#stock-symbo]s) folosind aceste simbolun precum API-ul de pretun tintä să trimitä datele prin socket cäte o companie (symbol) o datä la 3 secunde (pentru senalizare/deseriahzare se pot utiliza functiile loads 9i dumps din modulul json din Python, iar in Kotlin se poate utiliza kotlinx_seriahzation https://github.com/Kotlin/kotlinx.serialization _ De asemenea, se va implementa in Python un flux de date direct (direct stream) utilizänd framework-ul Apache Spark (PySpark); care să preia datele de la serverul TCP să le prelucreze astfel: - se va calcula pentru actiunile fiecärei companii profitul mediu care putea fi fäcut in luna curentä, pe baza preturilor estimate de anali>ti (targetMean targetLow); - se vor filtra rezultatele, pästrandu-le doar pe cele pentru care procentul de profit > 40% - pentru fiecare RDD in parte, se va afi>a compania (symbol) profitul mediu. Existä o limitare de 60 de apeluri / minut la fiecare cheie API_ Se poate crea un cont pe platforma finnhub io (Üs://finnhub.io/register), sau se poate utiliza urmätoarea cheie (token) API: br17eb7rh5re11vc07fg
8. Să se creeze în Python un server TCP care să facă o cerere pentru simbolurile companiilor cu acțiuni (vezi documentația: https://finnhub.io/docs/api#stock-symbols) și folosind aceste simboluri precum și API-ul de știri despre o companie (https://finnhub.io/docs/api#company-news) să trimită știrile din ziua curentă prin socket, câte o știre o dată la 3 secunde (pentru serializare/deserializare se pot utiliza funcțiile loads și dumps din modulul json din Python, iar în Kotlin se poate utiliza kotlinx.serialization

https://github.com/Kotlin/kotlinx.serialization). De asemenea, se va implementa în Kotlin un flux de date direct (direct stream) utilizând framework-ul Apache Spark, care să preia datele de la serverul TCP și să le prelucreze astfel: - se vor filtra acele știri a căror sursă este "Yahoo"; - se vor filtra acele știri al căror rezumat depășește 500 de caractere; - pentru fiecare RDD în parte, se va afișa URL-ul corespunzător știrii, data și titlul. Există o limitare de 60 de apeluri / minut la fiecare cheie API. Se poate crea un cont pe platforma finnhub.io (https://finnhub.io/register), sau se poate utiliza următoarea cheie (token) API: brmr2kfrh5rcss140jmg

9. Pornind de la aplicația exemplu din laboratorul 10, să se modifice aceasta astfel încât să se introducă un procesor de flux care să monitorizeze toate operațiile efectuate de celelalte procesoare de flux în sensul scrierii într-un jurnal local. Se vor utiliza microservicii Python și o comunicare prin RabbitMQ.
   Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului. Se vor respecta principiile SOLID pentru microservicii. La prezentare se va arăta diagrama de clase și se va explica aceasta împreună cu gradul de respectare a principiilor. Apoi va urma discuția pe cod.
   ---------- Rezolvare: Serviciu care fie pornit primul si sa primeasca prin rabbitMQ, pe o coada separata, mesaje de la toate serviciile care ruleaza si sa le puna intr-un fisier local.
10. Pornind de la exemplul 2 din laboratorul 6, să se modifice acesta astfel încât să utilizeze un fișier pe disc pe post de memorie cache (se va utiliza numai adăugare și analiza pe bază de marcaje temporale).
    Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului. Se va utiliza orchestrarea și respectarea principiilor SOLID pentru microservicii. La prezentare se va arăta diagrama de clase și se va explica aceasta împreună cu gradul de respectare a principiilor. Apoi va urma discuția pe cod.
11. Utilizând algoritmul Map-Reduce (implementarea cu stdin/stdout), să se calculeze indexul invers pentru un set de documente text. Funcția de mapare va parsa conținutul fiecărui document și va returna perechi de forma <word, {document_id: 1}>, unde document_id este numele documentului, iar 1 reprezintă o apariție a cuvântului respectiv. Funcția de reducere primește toate perechile, sortează perechile corespunzătoare aceluiași cuvânt și emite o pereche de forma <word, {document_id_i: count_word_in_document_id_i, ...}>.
12. Pornind de la exemplul 1 din laboratorul 6 să se modifice acesta astfel încât să fie câte un microserviciu separat pentru fiecare operație CRUD, iar acestea să fie accesate prin intermediul unui model de proiectare de tip poartă API în interiorul altui microserviciu. Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului. Se va utiliza orchestrarea și respectarea principiilor SOLID pentru microservicii. La prezentare se va arăta diagrama de clase și se va explica aceasta împreună cu gradul de respectare a principiilor. Apoi va urma discuția pe cod.
13. Pornind de la tema pe acasă care se referea la crearea unei aplicații REST pentru gestiunea cheltuielilor unei familii (laboratorul 4), să se realizeze următoarele:
    - să se deseneze diagrama de clase;
    - să se implementeze un serviciu (utilizând modelul porții API sau nu) care asigură o metodă de cache ce răspunde dacă noua cerere apare în primele 30 de minute;
    - să se implementeze un serviciu de replicare care multiplică serviciile respective și va combina toate cache-urile serviciilor replicate într-unul global;
    Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului. Se va utiliza orchestrarea și respectarea principiilor SOLID pentru microservicii. La prezentare se va arăta diagrama de clase și se va explica aceasta împreună cu gradul de respectare a principiilor. Apoi va urma discuția pe cod.
14. Pornind de la aplicația exemplu de chat din laboratorul 8, să se modifice aceasta prin utilizarea corutinelor și să se elimine procesorul central de mesaje. În locul lui să fie un procesor care să mențină o listă de utilizatori activi, iar apoi cei care doresc să comunice verifică dacă utilizatorul este activ, îi află adresa și portul, după care inițiază o comunicare punctuală.
    Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului. Se vor respecta principiile SOLID pentru microservicii. La prezentare se va arăta diagrama de clase și se va explica aceasta împreună cu gradul de respectare a principiilor. Apoi va urma discuția pe cod.

15. Utilizând algoritmul Map-Reduce (implementarea cu stdin/stdout), să se calculeze indexul invers pentru un set de URL-uri. Funcția de mapare va parsa conținutul fiecărei pagini web (se pot folosi modulele requests și BeautifulSoup) și va returna perechi de forma <word, {URL: 1}>, unde valoarea 1 reprezintă o apariție a cuvântului pe pagina respectivă. Funcția de reducere primește toate perechile, sortează perechile corespunzătoare aceluiași cuvânt și emite o pereche de forma <word, {URL_i: count_word_in_URL_i, ...}>.
16. Pornind de la aplicația exemplu pentru ciurul lui Eratostene din laboratorul 11, să se modifice aceasta astfel încât să se găsească (dacă există) perechile de numere care satisfac a * b = a + b * 3 (această verificare va fi făcută în funcție) unde a aparține lui A, b aparține lui B, iar A și B sunt două ADT-uri cu câte 100 de valori inițializate aleator (restul în programul principal). Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului.
17. Pornind de la aplicația exemplu pentru ciurul lui Eratostene din laboratorul 11, să se modifice aceasta astfel încât să se calculeze intersecția între A și B, unde A și B sunt două ADT-uri ce conțin fiecare câte 100 de valori inițializate aleator, apoi să se depună într-un ADT C rezultatele.
18. Pornind de la aplicația exemplu din laboratorul 10, să se modifice aceasta astfel încât să permită primirea de evaluari între 1-5 privind calitatea serviciului și să salveze numele utilizatorului și evaluarea într-un fișier local. Se vor utiliza microservicii Python și o comunicare prin RabbitMQ.
    Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului. Se vor respecta principiile SOLID pentru microservicii. La prezentare se va arăta diagrama de clase și se va explica aceasta împreună cu gradul de respectare a principiilor. Apoi va urma discuția pe cod.
19. Pornind de la aplicația exemplu din laboratorul 10, să se modifice aceasta astfel încât să se introducă un procesor de flux care să realizeze statistici cu privire la erorile (posibile surse: cele de comunicare, cele de la sistemul de cozi) și tipul acestora care au apărut (dacă au apărut) pentru o licitație până la adjudecare și apoi să scrie aceste informații într-un fișier local. Se vor utiliza microservicii Python și o comunicare prin RabbitMQ.
    Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului. Se vor respecta principiile SOLID pentru microservicii. La prezentare se va arăta diagrama de clase și se va explica aceasta împreună cu gradul de respectare a principiilor. Apoi va urma discuția pe cod.
20. Pornind de la aplicația exemplu din laboratorul 10, să se modifice aceasta astfel încât să se introducă un procesor de flux care să realizeze statistici cu privire la numărul de mesaje care au fost necesare pentru o licitație până la adjudecare și apoi să scrie aceste informații într-un fișier local. Se vor utiliza microservicii Python și o comunicare prin RabbitMQ.
    Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului. Se vor respecta principiile SOLID pentru microservicii. La prezentare se va arăta diagrama de clase și se va explica aceasta împreună cu gradul de respectare a principiilor. Apoi va urma discuția pe cod.
21. Pornind de la aplicația exemplu cu Okazii (simulat evident) din laboratorul 7, să se modifice aceasta astfel încât să se introducă un procesor de flux care să realizeze statistici cu privire la numărul de mesaje care au fost necesare pentru o licitație până la adjudecare și apoi să scrie aceste informații într-un fișier local.
    Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului. Se vor respecta principiile SOLID pentru microservicii. La prezentare se va arăta diagrama de clase și se va explica aceasta împreună cu gradul de respectare a principiilor. Apoi va urma discuția pe cod. +1.
22. Pornind de la aplicația exemplu cu Okazii (simulat evident) din laboratorul 7, să se modifice aceasta astfel încât să se introducă un procesor de flux care să permită împreună cu alte microservicii (care vor trebui eventual create) să se inițieze o discuție privată între câțiva utilizatori care licitează (deci ne mai trebuie cel puțin un procesor master și unul de comunicație replicat pentru fiecare utilizator - se va utiliza ori partea de bidding, ori partea de comunicație).
    Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului. Se vor respecta principiile SOLID pentru microservicii. La prezentare se va arăta diagrama de clase și se va explica aceasta împreună cu gradul de respectare a principiilor. Apoi va urma discuția pe cod.
23. Să se creeze un server TCP în Python care să facă o cerere la un API de știri (de exemplu: https://finnhub.io/docs/api#company-news) pentru o anumită companie (ex.: Apple - APPL) și să trimită știrile

din ultimele 2 săptămâni prin socket câte o știre o dată la 5 secunde și utilizând Spark Streaming și Spark RDD, să se creeze un stream direct ce va prelucra știrile astfel încât să realizeze o analiză statistică de sentimente pe baza a două fișiere text ce conțin cuvinte pozitive/negative (încărcate în aplicație ca RDD-uri) - ca la tema laboratorului 14. Rezultatul clasificării fiecărei știri (pozitiv/negativ/neutru) va fi afișat la consolă. Fișierele de cuvinte pot fi descărcate din arhiva cu exemple de la laboratorul 14 (http://mike.tuiasi.ro/labsd14.zip). Există o limitare de 60 de apeluri / minut la fiecare cheie API. Se poate crea un cont pe platforma finnhub.io (https://finnhub.io/register), sau se poate utiliza următoarea cheie (token) API: brmu4j7rh5r90ebn6irg

24. Pornind de la exemplul de proiectare și implementare a unei aplicații care primește informații meteo (adică utilizând și serviciile de acolo eventual cu modificări) să se realizeze următoarele:
    - să se proiecteze și să se implementeze un serviciu care va scrie într-un fișier (de orice format la alegere) informațiile recepționate.
    - să se aplice modelul de proiectare de tip filtru și a celui de tip dirijor în proiectarea și implementarea unui serviciu care extrage din fluxul de date informațiile legate de numelui orașului, temperatura minimă și maximă (utilizând serviciul anterior) și le va scrie în fișiere cu acest nume.
    Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului. Se va utiliza înlănțuirea și respectarea principiilor SOLID pentru microservicii. La prezentare se va arăta diagrama de clase și se va explica aceasta împreună cu gradul de respectare a principiilor. Apoi va urma discuția pe cod.
25. Pornind de la exemplul de proiectare și implementare a unei aplicații care primește informații meteo (adică utilizând și serviciile de acolo eventual cu modificări) să se aplice modelul de proiectare de tip API și astfel să se încapsuleze aplicația de laborator. Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului.
    Se va utiliza înlănțuirea și respectarea principiilor SOLID pentru microservicii. La prezentare se va arăta diagrama de clase și se va explica aceasta împreună cu gradul de respectare a principiilor. Apoi va urma discuția pe cod.
26. Să se creeze în Kotlin un server TCP care să facă o cerere (khttp) pentru simbolurile companiilor cu acțiuni (vezi documentația: https://finnhub.io/docs/api#stock-symbols) și folosind aceste simboluri precum și API-ul de știri despre o companie (https://finnhub.io/docs/api#company-news) să trimită știrile din ziua precedentă prin socket, câte o știre o dată la 3 secunde (pentru serializare/deserializare se pot utiliza funcțiile loads și dumps din modulul json din Python, iar în Kotlin se poate utiliza kotlinx.serialization https://github.com/Kotlin/kotlinx.serialization). De asemenea, se va implementa în Python un flux de date direct (direct stream) utilizând framework-ul Apache Spark (PySpark), care să preia datele de la serverul TCP și să le prelucreze astfel: - se vor filtra acele știri care folosesc o imagine PNG; - se vor filtra acele știri al căror URL nu depășește 80 de caractere; - pentru fiecare RDD în parte, se va afișa URL-ul corespunzător știrii, data și titlul. Există o limitare de 60 de apeluri / minut la fiecare cheie API. Se poate crea un cont pe platforma finnhub.io (https://finnhub.io/register), sau se poate utiliza următoarea cheie (token) API: brmrfu7rh5rcss140ogg
27. Pornind de la aplicația exemplu de chat din laboratorul 8, să se modifice aceasta prin utilizarea corutinelor și introducerea unui procesor de flux care monitorizează comunicația și înlocuiește automat orice cuvânt aflat într-un dicționar (încărcat dintr-un fișier txt) cu xxx (numărul de x este dat de numărul de caractere al cuvântului). Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului. Se vor respecta principiile SOLID pentru microservicii. La prezentare se va arăta diagrama de clase și se va explica aceasta împreună cu gradul de respectare a principiilor. Apoi va urma discuția pe cod.
28. Pornind de la aplicația exemplu de chat din laboratorul 8, să se modifice aceasta prin utilizarea corutinelor și introducerea unui procesor de flux care monitorizează comunicația și înlocuiește automat orice cuvânt aflat într-un dicționar (încărcat dintr-un fișier txt) cu xxx (numărul de x este dat de numărul de caractere al cuvântului). Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului. Se vor respecta principiile SOLID pentru microservicii. La prezentare se va arăta diagrama de clase și se va explica aceasta împreună cu gradul de respectare a principiilor. Apoi va urma discuția pe cod.
29. Pornind de la aplicația exemplu de chat din laboratorul 8, să se modifice aceasta astfel încât să se introducă un procesor de flux care să filtreze toată comunicația după niște reguli (de exemplu: se acceptă toate care vin din gama de porturi XX-YY), iar rezultatul se salvează într-un fișier local.

Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului. Se vor respecta principiile SOLID pentru microservicii. La prezentare se va arăta diagrama de clase și se va explica aceasta împreună cu gradul de respectare a principiilor. Apoi va urma discuția pe cod.

30. Pornind de la aplicația exemplu de chat din laboratorul 8, să se modifice aceasta prin utilizarea corutinelor și introducerea unui procesor de flux care permite încărcarea și trimiterea către un alt utilizator al unui fișier.
    Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului. Se vor respecta principiile SOLID pentru microservicii. La prezentare se va arăta diagrama de clase și se va explica aceasta împreună cu gradul de respectare a principiilor. Apoi va urma discuția pe cod.
31. Pornind de la aplicația exemplu de chat din laboratorul 8, să se modifice aceasta prin utilizarea corutinelor și introducerea unui procesor de flux care implementează multicast-ul (unul la mai mulți - trebuie să existe posibilitatea creării unui subgrup de studenți).
    Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului. Se vor respecta principiile SOLID pentru microservicii. La prezentare se va arăta diagrama de clase și se va explica aceasta împreună cu gradul de respectare a principiilor. Apoi va urma discuția pe cod
32. Pornind de la aplicația exemplu de chat din laboratorul 8, să se modifice aceasta prin utilizarea corutinelor și introducerea unui procesor de flux care implementează broadcast-ul (unul la toți) și unul care implementează multicast-ul (unul la mai mulți - trebuie să existe posibilitatea creării unui sublistă de profesori). Deci, aplicația suportă mai mulți profesori.
    Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului. Se vor respecta principiile SOLID pentru microservicii. La prezentare se va arăta diagrama de clase și se va explica aceasta împreună cu gradul de respectare a principiilor. Apoi va urma discuția pe cod.
33. Să se creeze în Python un server TCP care să facă o cerere pentru simbolurile companiilor cu acțiuni (vezi documentația: https://finnhub.io/docs/api#stock-symbols) și folosind aceste simboluri precum și API-ul de profil gratuit al unei companii (https://finnhub.io/docs/api#company-profile2) să trimită datele prin socket câte un profil o dată la 3 secunde (pentru serializare/deserializare se pot utiliza funcțiile loads și dumps din modulul json din Python, iar în Kotlin se poate utiliza kotlinx.serialization https://github.com/Kotlin/kotlinx.serialization). De asemenea, se va implementa în Kotlin un flux de date direct (direct stream) utilizând framework-ul Apache Spark, care să preia datele de la serverul TCP și să le prelucreze astfel: - se vor filtra acele companii care sunt listate pe exchange-ul "NEW YORK STOCK EXCHANGE, INC."; - se vor filtra acele companii cu un IPO (initial public offer) din anul 2015 până în prezent; - pentru fiecare RDD în parte, se va afișa numele companiei, numărul de telefon și piața de capital (marketCapitalization). Există o limitare de 60 de apeluri / minut la fiecare cheie API. Se poate crea un cont pe platforma finnhub.io (https://finnhub.io/register), sau se poate utiliza următoarea cheie (token) API: brmr7v7rh5rcss140lq0
34. Să se creeze în Kotlin un server TCP care să facă o cerere (khttp) pentru simbolurile companiilor cu acțiuni (vezi documentația: https://finnhub.io/docs/api#stock-symbols) și folosind aceste simboluri precum și API-ul de știri despre o companie (https://finnhub.io/docs/api#company-news) să trimită știrile din ziua precedentă prin socket, câte o știre o dată la 3 secunde (pentru serializare/deserializare se pot utiliza funcțiile loads și dumps din modulul json din Python, iar în Kotlin se poate utiliza kotlinx.serialization
    https://github.com/Kotlin/kotlinx.serialization). De asemenea, se va implementa în Python un flux de date direct (direct stream) utilizând framework-ul Apache Spark (PySpark), care să preia datele de la serverul TCP și să le prelucreze astfel:
    - se vor filtra acele știri care folosesc o imagine PNG;
    - se vor filtra acele știri al căror URL nu depășește 80 de caractere;
    - pentru fiecare RDD în parte, se va afișa URL-ul corespunzător știrii, data și titlul. Există o limitare de 60 de apeluri / minut la fiecare cheie API. Se poate crea un cont pe platforma finnhub.io (https://finnhub.io/register), sau se poate utiliza următoarea cheie (token) API: brmrfu7rh5rcss140ogg
35. Utilizând algoritmul Map-Reduce (implementarea cu stdin/stdout), să se calculeze indexul invers pentru un set de documente text. Funcția de mapare va parsa conținutul fiecărui document și va returna perechi de forma <word, {document_id: 1}>, unde document_id este numele documentului, iar 1 reprezintă o apariție a

cuvântului respectiv. Funcția de reducere primește toate perechile, sortează perechile corespunzătoare aceluiași cuvânt și emite o pereche de forma <word, {document_id_i: count_word_in_document_id_i, ...}>.

36. Utilizând algoritmul Map-Reduce (implementarea cu stdin/stdout), să se calculeze indexul invers pentru un set de URL-uri. Funcția de mapare va parsa conținutul fiecărei pagini web (se pot folosi modulele requests și BeautifulSoup) și va returna perechi de forma <word, {URL: 1}>, unde valoarea 1 reprezintă o apariție a cuvântului pe pagina respectivă. Funcția de reducere primește toate perechile, sortează perechile corespunzătoare aceluiași cuvânt și emite o pereche de forma <word, {URL_i: count_word_in_URL_i, ...}>.
37. Pornind de la aplicația exemplu pentru ciurul lui Eratostene din laboratorul 11, să se modifice aceasta astfel încât să se găsească (dacă există) perechile de numere care satisfac a * b = a + b * 3 (această verificare va fi făcută în funcție) unde a aparține lui A, b aparține lui B, iar A și B sunt două ADT-uri cu câte 100 de valori inițializate aleator (restul în programul principal). Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului.
38. Pornind de la aplicația exemplu de chat din laboratorul 8, să se modifice aceasta prin utilizarea corutinelor și introducerea unui procesor de flux care implementează multicast-ul (unul la mai mulți - trebuie să existe posibilitatea creării unui subgrup de studenți). Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului. Se vor respecta principiile SOLID pentru microservicii. La prezentare se va arăta diagrama de clase și se va explica aceasta împreună cu gradul de respectare a principiilor. Apoi va urma discuția pe cod
39. Pornind de la exemplul 1 din laboratorul 6 să se modifice acesta astfel încât să fie câte un microserviciu separat pentru fiecare operație CRUD, iar acestea să fie accesate prin intermediul unui model de proiectare de tip poartă API în interiorul altui microserviciu. Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului. Se va utiliza orchestrarea și respectarea principiilor SOLID pentru microservicii. La prezentare se va arăta diagrama de clase și se va explica aceasta împreună cu gradul de respectare a principiilor. Apoi va urma discuția pe cod.
40. Pornind de la aplicația exemplu din laboratorul 10, să se modifice aceasta astfel încât să se introducă un procesor de flux care să realizeze statistici cu privire la erorile (posibile surse: cele de comunicare, cele de la sistemul de cozi) și tipul acestora care au apărut (dacă au apărut) pentru o licitație până la adjudecare și apoi să scrie aceste informații într-un fișier local. Se vor utiliza microservicii Python și o comunicare prin RabbitMQ. Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului. Se vor respecta principiile SOLID pentru microservicii. La prezentare se va arăta diagrama de clase și se va explica aceasta împreună cu gradul de respectare a principiilor. Apoi va urma discuția pe cod.
41. Pornind de la exemplul de implementare a agendei cu servicii REST din laborator să se realizeze următoarele: - să se deseneze diagrama de clase; - să se implementeze un serviciu (utilizând modelul porții API sau nu) care asigură o metodă de cache ce răspunde dacă noua cerere apare în primele 30 de minute; Dacă se consideră necesar se mai pot crea oricâte servicii se dorește conform gândirii studentului. Se va utiliza orchestrarea și respectarea principiilor SOLID pentru microservicii. La prezentare se va arăta diagrama de clase și se va explica aceasta împreună cu gradul de respectare a principiilor. Apoi va urma discuția pe cod.