UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Thesis Matcher

Aplicație de repartizare a studenților la profesori coordonatori de licență

propusă de

Andrei Dascălu

Sesiunea: februarie, 2023

Coordonator științific

Lect. Dr. Frăsinaru Cristian

UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

Thesis Matcher

Andrei Dascălu

Sesiunea: februarie, 2023

Coordonator științific

Lect. Dr. Frăsinaru Cristian

Avizat,
Îndrumător lucrare de licență,
Lect. Dr. Frăsinaru Cristian.
Data: ..... Semnătura: .....

Declarație privind originalitatea conținutului lucrării de licență

Subsemnatul **Dascălu Andrei** domiciliat în **România, jud. Iași, mun. Iași, strada Sf. Teodor nr. 14**, născut la data de **03 noiembrie 2000**, identificat prin CNP **5001103226752**, absolvent al Facultății de informatică, **Facultatea de informatică** specializarea **informatică**, promoția 2022, declar pe propria răspundere cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art. 143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul **Thesis Matcher** elaborată sub îndrumarea domnului **Lect. Dr. Frăsinaru Cristian**, pe care urmează să o susțin în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului ei într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diplomă sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Data: ..... Semnătura: .....

Declarație de consimțământ

Prin prezenta declar că sunt de acord ca lucrarea de licență cu titlul **Thesis Matcher**, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test, etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de informatică.

De asemenea, sunt de acord ca Facultatea de informatică de la Universitatea ”Alexandru-Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Absolvent **Andrei Dascălu**

Data: ..... Semnătura: .....

Cuprins

Motivație

Lucrarea de licență reprezintă culminarea anilor de facultate, o dovadă a studentului de deprindere a unor noțiuni, de specializare într-un anumit domeniu și de capabilitatea a acestuia de a contribui cu soluții proprii la problemele curente sau viitoare ale societății. În consecință, alegerea unei tematici adecvate pentru lucrarea de licență este un pas de bază datorită unor motive pertinente. În primul rând, abordarea unei tematici cât mai aproape de domeniile sau subiectele de interes ale studentului rezultă într-o atenție mai mare și o implicare corespunzătoare ale acestuia în realizarea în sine a lucrării. În al doilea rând, există o legătură directă între eficiența și temeinicia realizări lucrării de licență și colaborarea dintre profesorul coordonator și student, fiind necesară o comunicare constantă și productivă prind feedback-ul în ambele sensuri, dar și prin resurse, materiale sau idei.

Profesorii coordonatori au un număr de locuri limitat, prin urmare este necesară o repartizare eficientă și mai ales optimă a studenților în funcție de preferințe, luând în același timp în calcul și punctele de vedere ale profesorilor. De asemenea, studenții ar trebui să aibă posibilitatea să cunoască dinainte tematicile propuse de profesori și eventualele condiții prealabile de realizare a respectivelor teme. O aplicație centralizată este așadar întrutotul necesară optimizării și îmbunătățirii în general ale acestui proces organizatoric din cadrul facultății.

Introducere

0.0.1 Scopul documentului

Documentul de față are ca scop prezentarea problemei și a unei soluții propuse adecvate acesteia. În acest fel, se urmărește descompunerea problemei în subprobleme și detalierea acestora, ilustrarea unor soluții deja existente, analizarea aplicației web propuse, în mod sistematic, evidențiind atât detalii de abordare, arhitectură, tehnologii utilizate, implementare.

0.0.2 Scopul aplicației

Aplicația *Thesis Matcher* este o soluție web ce urmărește să rezolve problema alocării studenților din anii terminali la profesorii coordonatori.

Unul dintre principalele obiective este centralizarea întregului proces, de la repartizare până la alte colaborare și alte aspecte organizatorice, asigurând astfel o desfășurare bună și mai sigură. Fiind un proces anual, cu un număr semnificativ de părți implicate, este inevitabilă necesitatea de a automatiza într-o anumită măsură desfășurarea acestuia.

Un obiectiv de asemenea important este simplificarea obținerii informaților de interes, atât de către studenți, cât și de profesori. Studenții au posibilitatea să afle tematicile propuse de fiecare coordonator, eventual să propună o idee proprie. Participanții pot urmări în timp real situația locurilor disponibile, precum și alte statistici.

Cel de al treilea obiectiv este optimizarea repartizării prin implementarea ierarhizării de către student a opțiunilor alese sub formă de preferințe.

0.0.3 Modul de implementare

Soluția se prezintă sub formă de aplicație web ce se împarte în două ”sub-aplicații”.

Partea de *Front-End* este implementată în Angular și găzduită pe un server Firebase. Partea de *Back-End* este implementată în Java Spring Boot și găzduită utilizând serviciile Google Cloud. Această parte cuprinde atât comunicarea cu Front-End-ul prin intermediul request-urilor, dar și procesare sub forma implementării unui algoritm de stable-matching (stable-marriage) adecvat problemei.

Capitolul 1

Teorii și tehnologii utilizate

În acest capitol sunt prezentate în primul rând tehnologiile utilizate în implementarea aplicației. În cea de a doua parte este evidențiat în mare parte particularități ale algoritmicii utilizate în determinarea unei alocări optime specifice problemei, în alte cuvinte algoritmul de *stable matching* (*stable marriage*).

1.1 Angular

Angular este un framework de JavaScript scris în TypeScript și menținut de Google. Framework-ul a fost dezvoltat în principal pentru crearea aplicațiilor web *single-page*, într-o manieră ce ușurează mentenanța și dezvoltarea ulterioară.

1.1.1 Scurt istoric

În 2010, Miško Hevery, un angajat la Google la acel timp, a lansat un proiect cu numele *AngularJS* care a fost apreciat în mare măsură de comunitate. Între anii 2014-2015 a avut loc o reîmprospătare majoră a framework-ului însemnând de fapt o rescriere majoră a acestuia. Noua versiune avea să fie numită simplu Angular. Au urmat câțiva ani de tranziție deoarece multe proiecte deja în producție erau utilizau AngularJS și trebuiau refactorizate. În acest moment, Angular este cel mai folosit framework

de front-end, în specialde dezvoltatorii de la Google și de către start-up-uri. Exemple de companii recunoscute ce folosesc Angular sunt Microsoft, Gmail, PayPal, Forbes.

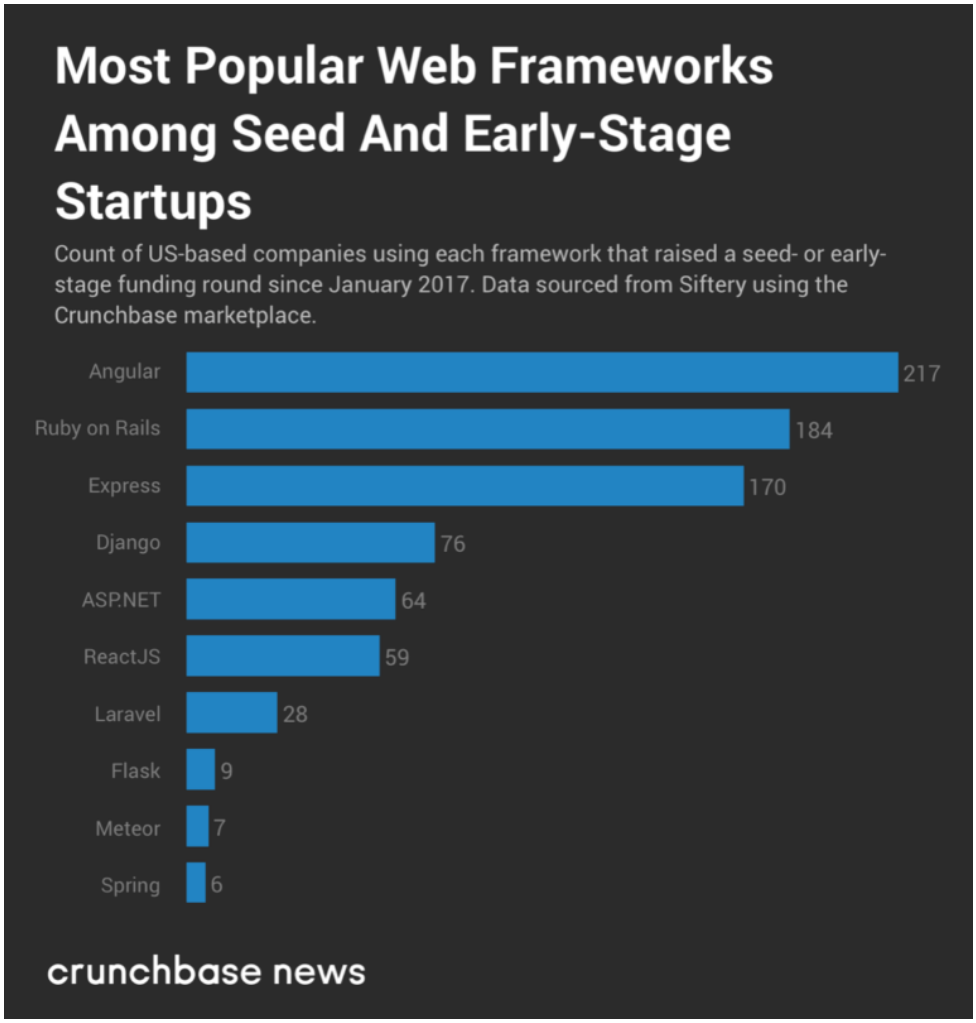

Figura 1.1: `https://www.moveoapps.com/blog/how-to-use-angular-development/`

1.1.2 Particularități

Motivul alegerii Angular pe partea de front-end este reprezentat de o serie de caracteristici.

Un astfel de motiv este *arhitectura bazată pe componenta*, fiind astfel o formă de programare orientată pe obiect. Utilizatorul crează în mod uzual clase corespunzătoare componentelor ce conțin și un șablon HTML (*eng.* HTML template). Pentru simplificare, Angular oferă și opțiunea de ”injectare” a serviciilor *custom* sau *in-built* într-o componentă ce utilizează aceste funcționalități. În acest fel, utilizatorul poate reutiliza, înlocui, modifica componente în diverse locuri, obținâându-se astfel un UI (User

Interface) modularizat.

Un alt motiv este modul de încărcare a paginii web. Angular folosește *lazy loading* ce permite încărcarea instantanee a website-urilor, prin afișarea doar a componentelor cerute și necesare utilizatorului, în timp ce celelalte sunt pregătite în fundal pentru alte eventualități.

Dependency injection reprezintă un al treilea motiv, un design pattern ce permite împărțirea lucrului între diferite servicii, distribuind în mod eficient sarcinile. Prin inițializarea dependențelor, Angular reușește să reducă în mod considerabil codul de tip *boilerplate* (fragmente similare de cod des utilizat între care există mici diferențe) și să extindă mai ușor o astfel de aplicație.

Framework-ul are trei tipuri de *dependency injections*:

1. Constructor injection
2. Setter injection
3. Interface injection

Din punct de vedere al arhitecturii, Angular este în totalitate un framework MVC (model-view-controller), după cum se observă în figura următoare.

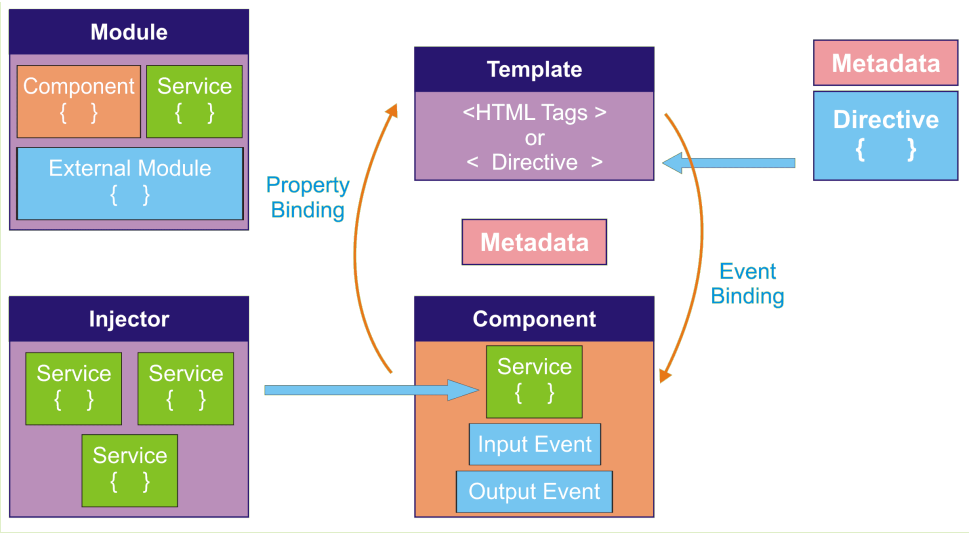

Figura 1.2: `https://www.ngdevelop.tech/wp-content/uploads/2017/12/Angular_Architecture.png`

1.2 Spring Boot

Spring Boot este un micro-framework open-source folosit pentru a crea aplicații Spring cu microservicii. Spre deosebire de alte framework-uri de Java, acesta oferă configurări XML flexibile, procesare în loturi puternică, tranzacții cu baza de date și o varietate de instrumente de dezvoltare.

1.2.1 Scurt istoric

Framework-ul *Spring* a fost creat în 2004 pentru a simplifica dezvoltarea programelor pe partea de server. În aprilie 2014 a fost lansat Spring boot 1.0.0 în urma unor cereri din partea programatarilor de a configura serviciile de web container într-un container spring din metoda principală. În decembrie 2016 a fost lansat Spring Boot 1.3 ulterior trecerii framework-ului Spring de la versiunea 4.1 la 4.2 și includea sprinjin pentru fisiere JAR complet executabile, noi utilitare spring-boot-dev și auto-configurare pentru tehnologii de caching.

1.2.2 Motivație

Spring Boot este bazat pe Java, unul dintre cele mai populare limbaje de programare. Framework-ul are o comunitate vastă de utilizatori cu diverse materiale și cursuri.

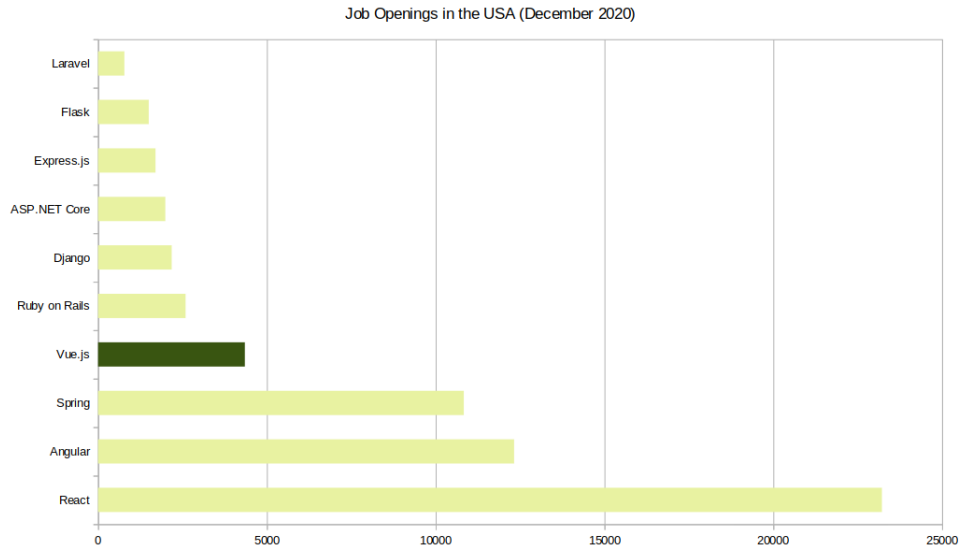

Figura 1.3: `https://miro.medium.com/max/1400/1*Y6A_rS5IdRDUG4KRIFP_fA.png`

Avantajele prinicipale sunt după cum urmează. Spring Boot este *multi-threaded*, util pentru operații repetitive și de durată. Facilitează crearea și testarea aplicațiilor Java oferind un setup default pentru unit și integration testing. Există de asemenea posibilitatea de a integra Spring Boot cu ecosistemul Spring ce include Spring Data, Spring Security, Spring ORM și Spring JDBC într-un mod simplificat.

1.2.3 Particularități

Principala particularitate a Spring Boot o reprezintă adnotările (*Spring Boot annotations*) utilizate în auto configurare. De exemplu, **@SpringBootApplication** marchează metoda principală a aplicației și este obligatorie. **@EnableAutoConfiguration** oferă oricărei clase pe care o marchează cu opțiunea de Automatic Configuration. **@ComponentScan** scanează la inițializare toate declarările de *beans* și pachete.

Un alt detaliu este utilizarea *Spring Starter Dependencies* ce facilitează gestionarea dependențelor unei aplicații în continuă dezvoltare.

Spring Boot Actuator oferă utilitare de producție aplicației. Actuator este utilizat în mare parte pentru a obține informații de funcționare despre o aplicație în rulare (metrice, info, dump, env, etc.), cu ajutorul HTTp endpoints și JMX beans. Ultima versiune, Spring Boot 2.x Actuator, suportă și modele CRUD.

1.3 Cloud Firestore

Cloud Firestore este un serviciu oferit de Firebase și este o bază de date în cloud, NoSQL, pentru a reține și sincroniza date între dispozitive folosind *listeners* în timp real. Firestore oferă de altfel și suport pentru o integrare facilă cu serviciile Google Cloud.

1.3.1 Particularități

Fiind un model NoSQL, Firestore reține datele sub formă de *documente* ce conțin câmpuri mapate la valori. Documentele formează la rândul lor diferite *colecții*, containere folosite pentru organizarea datelor și crearea ușoară a cererilor (queries). Documentele pot reprezenta un număr mare de tipuri de date, de la string-uri până la tipuri complexe precum anumite obiecte. De asemenea, orice instanță, document, are posibilitatea să aibă subcolecții. Spre exemplu, în cazul de față, orice student ar putea avea drept subcolecție o listă de profesori reprezentând preferințele sale.

1.3.2 Avantaje

Cererile în cazul Cloud Firestore pentru crearea și primirea de date sunt expresive, flexibile și eficiente. Datele pot fi primite ușor sub forme de documente, iar posibilitatea de creare a subcolecțiilor ajută după caz la evitarea operațiilor costisitoare de join.

De asemenea, există opțiuni de sortare, filtrare și paginare a datelor primite.

Un avantaj important este reprezentat de *realtime listeners* ce permit obținerea numai a informațiilor nou apărute, fără a extrage întreaga colecție de fiecare dată.

1.3.3 Aplicabilitate

Cloud Firestore are rolul în cazul acestei aplicații în principal de a reține participanții la procesul de repartizare, adică utilizatorii reprezentați de studenți și profesori, sub formă de documente. Fiecare document are o subcolecție de obiecte corespunzătoare

reprezentând preferințele sale. Este de menționat că această subcolecție este sortată, fiind absolut necesară cunoașterea ierarhiei de opțiuni. Pe lânga preferințe, utilizatorii au și câmpuri specifice unei astfel de aplicații web, precum *username*, *email*, date administrative.

1.4 Algoritmica

1.4.1 Problema stable matching (SMP)

Problema constă în găsirea unei potriviri stabile (stable match) între două mulțimi de aceași dimensiune de elemente, luând în considerare o ierarhie a preferințelor. Astfel, o soluție este o bijecție între elementele primei mulțimi și cele din cea de a doua. O potrivire nu este stabilă dacă:

1. Există un element α în mulțimea întâi care preferă un anumit element β din cea de a doua mulțime în detrimentul unui element din a doua mulțime deja cuplat.

2. β preferă α în detrimentul elementului cu care β deja este cuplat.

Cea mai întâlnită formă a problemei este *stable marriage problem*. Fie *n* bărbați și *n* femei, fiecare persoană având o ierarhizare a tuturor membrilor de gen opus în funcție de preferințe.

Prima soluție a fost prezentată în 1962 de către David Gale și Lloyd Shapley care au demonstrat că pentru orice număr de participanți, există o soluție pentru a face toate cuplurile stabile.

1.4.2 Algoritmul Gale-Shapley

Algoritmul implică un număr de iterații. În prima iterație, fiecare element din prima mulțime A încearcă cuplarea cu prima sa obțiune din mulțimea B. Elementele din B sunt cuplate inițial cu elementul pretendent pe care îl preferă cel mai mult.

În iterațiile ulterioare, fiecare element din A necuplat încă încearcă să se cupleze cu următoarea opțiune din preferințele (necuplată sau nu). Elementele din B pot schimba partenerul curent cu cel propus în cazul în care este deja cuplată dacă îl preferă pe cel din urmă.

Procesul se repetă până când există o bijecție între cele două mulțimi.

1.4.3 Instanța problemei prezente

În cazul problemei distribuirii studenților la profesorii coordonatori de licență, fiecare student primind implicit o temă unică de licență, există atât asemănări cu *SMP* (Stable Marriage Problem), cât și o serie de deosebiri particulare care necesită o adaptare a algoritmului de rezolvare.

Asemănări

Există două mulțimi ce trebuie cuplate într-un mod stabil. Cuplajul este realizat tinând cont și în această instanță de o serie de preferințe. Ambele tipuri de participanți au o ierarhizare a preferințelor sale. Din punct de vedere al temelor de licență, cel mai probabil există o relație de bijecție între studenți și tematicile primite sau propuse și acceptate.

Deosebiri

O primă deosebire observată este că mulțimile nu au un număr egal de elemente. În mod natural, mulțimea studenților este semnificativ mai mare decât cea a profesorilor.

În al doilea rând, relația dintre cele două mulțimi nu este de bijecție, ci mai degrabă una injectivă. Fiecare student trebuie să aibă la finalul procesului de repartizare atribuit un profesor coordonator de licență. Cu alte cuvinte,

$$\forall\, stud,\ unde\ stud \in Studs,\ \exists\, prof \in Profs,$$

$$BTD(stud) = prof$$

unde *BTD* = Bachelor Thesis Distributor, funcție ce repartizează fiecare student unui profesor, *Studs* = mulțimea studenților, *Profs* = mulțimea profesorilor.

În consecință, este inevitabil ca un profesor să aibă sub coordonare mai mulți studenți, o altă particularitate fiind însă că un profesor are o limită prestabilite de astfel de studenți.

La finalul procesului de repartizare, studenții care nu au o preferință sunt repartizați aleatoriu sau după un anumit criteriu arbitrar profesorilor care mai au locuri libere.

Capitolul 2

Titlul celui de-al doilea capitol

Facilisi nullam vehicula ipsum a arcu. Purus semper eget duis at tellus at. Adipiscing tristique risus nec feugiat. Eu volutpat odio facilisis mauris sit. Porta nibh venenatis cras sed. Penatibus et magnis dis parturient. Sollicitudin aliquam ultrices sagittis orci a. Senectus et netus et malesuada fames ac turpis egestas integer. Cras tincidunt lobortis feugiat vivamus at augue eget arcu dictum. Leo vel fringilla est ullamcorper eget nulla facilisi etiam dignissim. Nulla aliquet enim tortor at auctor urna nunc id cursus. Elit duis tristique sollicitudin nibh. Sagittis nisl rhoncus mattis rhoncus urna neque viverra. Convallis posuere morbi leo urna molestie at. Quisque egestas diam in arcu cursus euismod.

2.1 Titlul secțiunii 1

A diam sollicitudin tempor id eu nisl. Hac habitasse platea dictumst vestibulum. Integer enim neque volutpat ac tincidunt. Facilisi nullam vehicula ipsum a arcu cursus vitae congue. Vel turpis nunc eget lorem. Vestibulum mattis ullamcorper velit sed ullamcorper morbi tincidunt ornare. Nunc sed blandit libero volutpat. Sit amet luctus venenatis lectus magna fringilla urna porttitor. Hac habitasse platea dictumst quisque sagittis purus. Sed faucibus turpis in eu mi bibendum neque egestas. Vel orci porta non pulvinar neque laoreet suspendisse interdum consectetur. Erat nam at lectus urna duis convallis convallis tellus id. Tristique sollicitudin nibh sit amet commodo nulla facilisi nullam vehicula. Etiam dignissim diam quis enim lobortis scelerisque. Nunc congue nisi vitae suscipit tellus mauris a diam maecenas. Lacus viverra vitae congue eu consequat ac felis donec. Mauris sit amet massa vitae tortor condimentum. Mauris

augue neque gravida in. Lorem ipsum dolor sit amet. Arcu dui vivamus arcu felis bibendum ut tristique et.

2.2 Titlul secțiunii 2

Sit amet mauris commodo quis imperdiet massa tincidunt nunc pulvinar. Ligula ullamcorper malesuada proin libero nunc consequat interdum. Mauris a diam maecenas sed enim ut. Ut sem nulla pharetra diam sit amet nisl suscipit adipiscing. Leo duis ut diam quam nulla. Neque ornare aenean euismod elementum. Vitae sapien pellentesque habitant morbi tristique senectus. Lectus magna fringilla urna porttitor rhoncus dolor purus non enim. Egestas sed sed risus pretium quam vulputate dignissim suspendisse in. At quis risus sed vulputate odio ut enim. Hac habitasse platea dictumst quisque sagittis. Lectus vestibulum mattis ullamcorper velit sed. Massa vitae tortor condimentum lacinia quis vel eros donec ac. Vulputate dignissim suspendisse in est ante. Sed faucibus turpis in eu mi bibendum neque. Enim eu turpis egestas pretium aenean pharetra magna. Tellus mauris a diam maecenas.

2.3 Titlul secțiunii 3

Faucibus ornare suspendisse sed nisi lacus sed. Mi in nulla posuere sollicitudin aliquam ultrices. Lacus suspendisse faucibus interdum posuere lorem ipsum dolor sit amet. Odio tempor orci dapibus ultrices in iaculis nunc sed augue. Congue eu consequat ac felis donec et odio. Enim ut sem viverra aliquet eget sit amet. Sit amet consectetur adipiscing elit duis tristique sollicitudin. Quis blandit turpis cursus in. Cras fermentum odio eu feugiat pretium nibh ipsum consequat nisl. Non curabitur gravida arcu ac tortor dignissim convallis aenean. Porta non pulvinar neque laoreet suspendisse interdum consectetur libero id. Lacus viverra vitae congue eu consequat ac felis. Vulputate dignissim suspendisse in est ante in nibh mauris. Amet mauris commodo quis imperdiet massa. Varius sit amet mattis vulputate enim nulla aliquet. Pellentesque diam volutpat commodo sed egestas egestas. Amet est placerat in egestas erat imperdiet sed euismod. Scelerisque varius morbi enim nunc faucibus a pellentesque sit. Ut sem viverra aliquet eget sit amet tellus cras. Sem integer vitae justo eget magna fermentum iaculis eu.

Capitolul 3

Titlul celui de-al treilea capitol

Amet venenatis urna cursus eget. Quam vulputate dignissim suspendisse in est ante. Proin nibh nisl condimentum id. Egestas maecenas pharetra convallis posuere morbi. Risus viverra adipiscing at in. Vulputate eu scelerisque felis imperdiet. Cras adipiscing enim eu turpis egestas pretium aenean pharetra. In aliquam sem fringilla ut morbi tincidunt augue. Montes nascetur ridiculus mus mauris. Viverra accumsan in nisl nisi scelerisque eu ultrices vitae. In nibh mauris cursus mattis molestie a iaculis. Interdum consectetur libero id faucibus nisl tincidunt eget. Gravida in fermentum et sollicitudin ac orci. Suscipit adipiscing bibendum est ultricies. Etiam non quam lacus suspendisse. Leo urna molestie at elementum eu facilisis sed odio morbi. Egestas congue quisque egestas diam in arcu cursus. Amet consectetur adipiscing elit ut aliquam purus.

3.1 Titlul secțiunii 1

Eros donec ac odio tempor. Facilisi morbi tempus iaculis urna id volutpat. Faucibus in ornare quam viverra orci sagittis eu. Amet tellus cras adipiscing enim eu turpis egestas. Integer feugiat scelerisque varius morbi. Platea dictumst vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Bibendum arcu vitae elementum curabitur. Eu nisl nunc mi ipsum faucibus. Id aliquet lectus proin nibh nisl condimentum id venenatis a. Cras adipiscing enim eu turpis egestas pretium. Quisque non tellus orci ac auctor augue mauris augue. Malesuada pellentesque elit eget gravida cum. Ut lectus arcu bibendum at. Massa id neque aliquam vestibulum morbi blandit. Posuere ac ut consequat semper viverra nam. Viverra adipiscing at in tellus integer feugiat

scelerisque varius morbi. Morbi enim nunc faucibus a pellentesque sit amet porttitor eget. Eu feugiat pretium nibh ipsum consequat nisl vel. Nisl purus in mollis nunc sed.

3.2 Titlul secțiunii 2

Elementum sagittis vitae et leo duis ut diam quam nulla. Purus sit amet volutpat consequat mauris nunc. Tincidunt augue interdum velit euismod in pellentesque massa. Nunc sed augue lacus viverra vitae congue. Porttitor leo a diam sollicitudin. Faucibus pulvinar elementum integer enim. Adipiscing bibendum est ultricies integer quis auctor elit. Blandit aliquam etiam erat velit scelerisque in. A iaculis at erat pellentesque adipiscing commodo elit at. Erat nam at lectus urna duis. Consequat ac felis donec et. Fermentum posuere urna nec tincidunt praesent semper feugiat nibh sed. Proin gravida hendrerit lectus a. Pretium viverra suspendisse potenti nullam ac tortor vitae purus. Arcu cursus euismod quis viverra nibh cras pulvinar mattis. Gravida arcu ac tortor dignissim convallis aenean. Quam nulla porttitor massa id neque aliquam vestibulum morbi. Sed viverra ipsum nunc aliquet. Quis enim lobortis scelerisque fermentum dui faucibus in.

Concluzii

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Nunc mattis enim ut tellus elementum sagittis vitae et. Placerat in egestas erat imperdiet sed euismod. Urna id volutpat lacus laoreet non curabitur gravida. Blandit turpis cursus in hac habitasse platea. Eget nunc lobortis mattis aliquam faucibus. Est pellentesque elit ullamcorper dignissim cras tincidunt lobortis feugiat. Viverra maecenas accumsan lacus vel facilisis volutpat est. Non odio euismod lacinia at quis risus sed vulputate odio. Consequat ac felis donec et odio pellentesque diam volutpat commodo. Etiam sit amet nisl purus in. Tortor condimentum lacinia quis vel eros donec. Phasellus egestas tellus rutrum tellus pellentesque eu tincidunt. Aliquam id diam maecenas ultricies mi eget mauris pharetra. Enim eu turpis egestas pretium.

Bibliografie

- Author1, *Book1*, 2018
- Author2, *Boook2*, 2017